STRIVE: Software Technology - Research Integration and Verification Environment

MoBIES Principal Investigators Meeting

Jonathan D. Preston

Lockheed Martin Aeronautics Company

July 24–26, 2002

Approved for Public Release, Distribution Unlimited

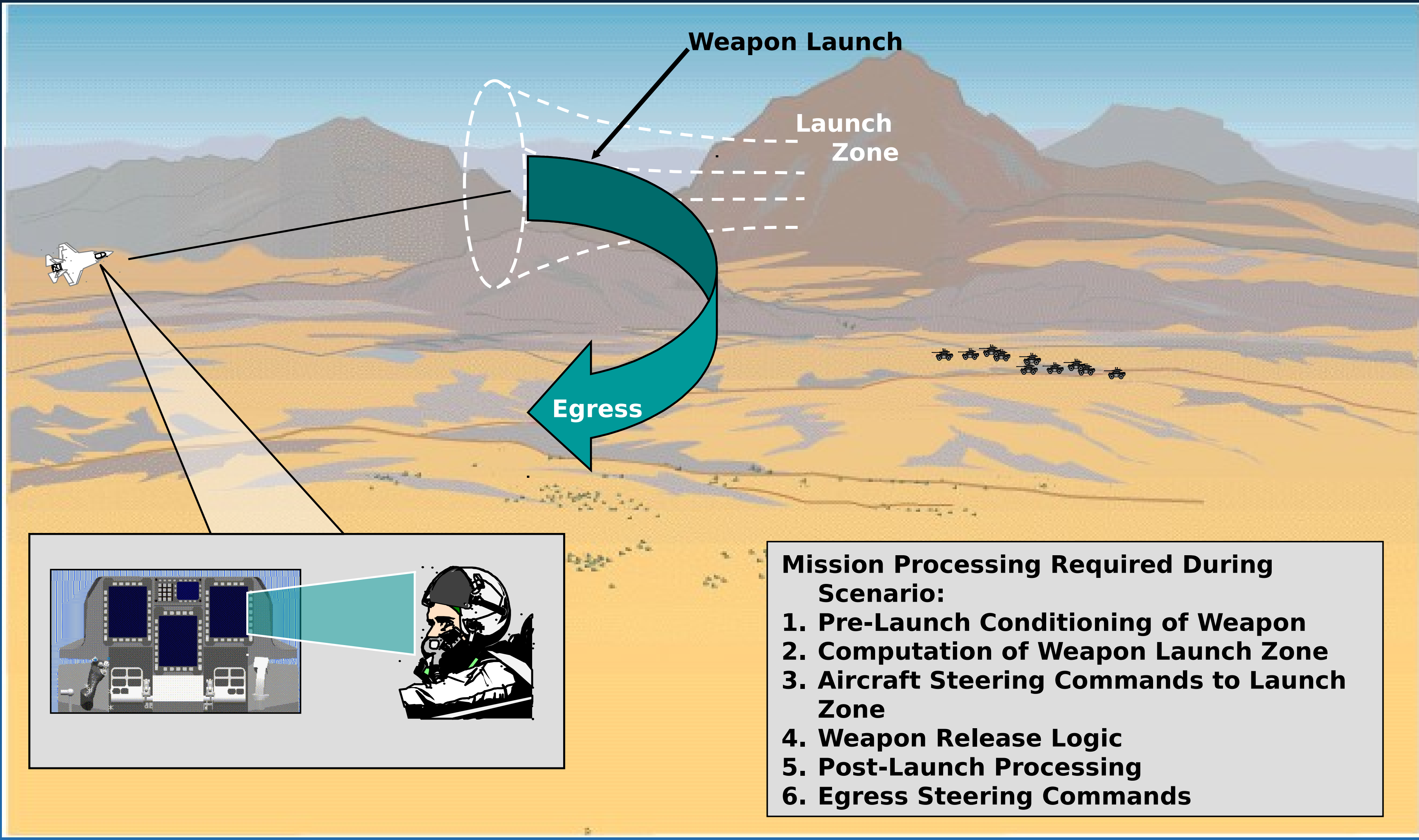
Subcontractors and Collaborators

- Carnegie Mellon University – TimeWeaver tool being used by STRIVE engineers in continuing experiments

- Vanderbilt – Potential collaboration, ongoing exploration

- Other potential collaborations possible in formal methods area (Kestrel, CMU, SRI)

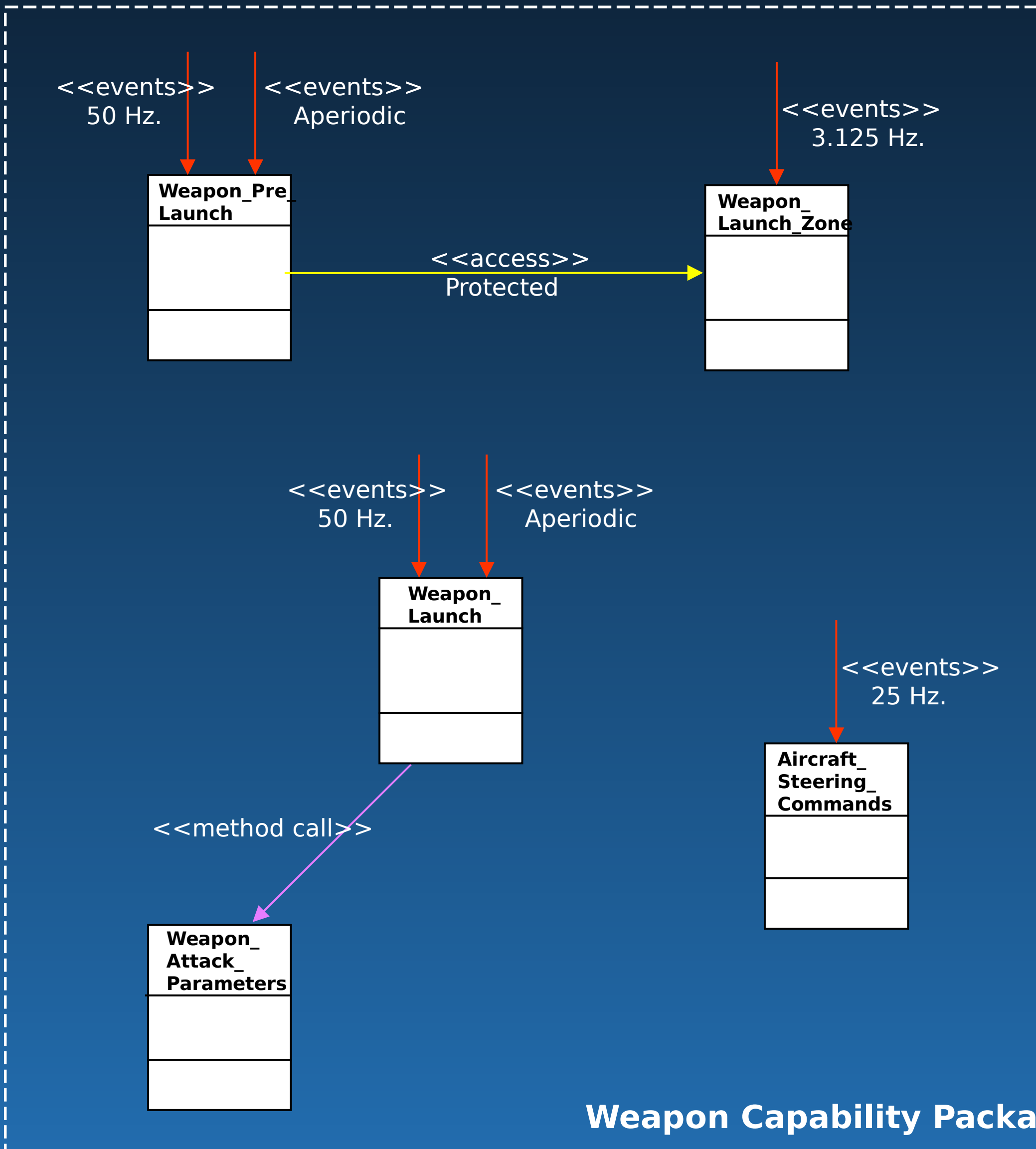
Problem Description – Experimental Application



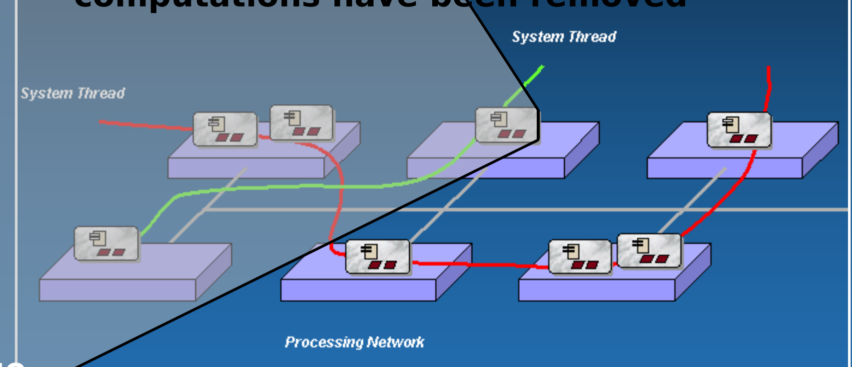
Mission Processing Required During Scenario:

1. Pre-Launch Conditioning of Weapon
2. Computation of Weapon Launch Zone
3. Aircraft Steering Commands to Launch Zone
4. Weapon Release Logic
5. Post-Launch Processing
6. Egress Steering Commands

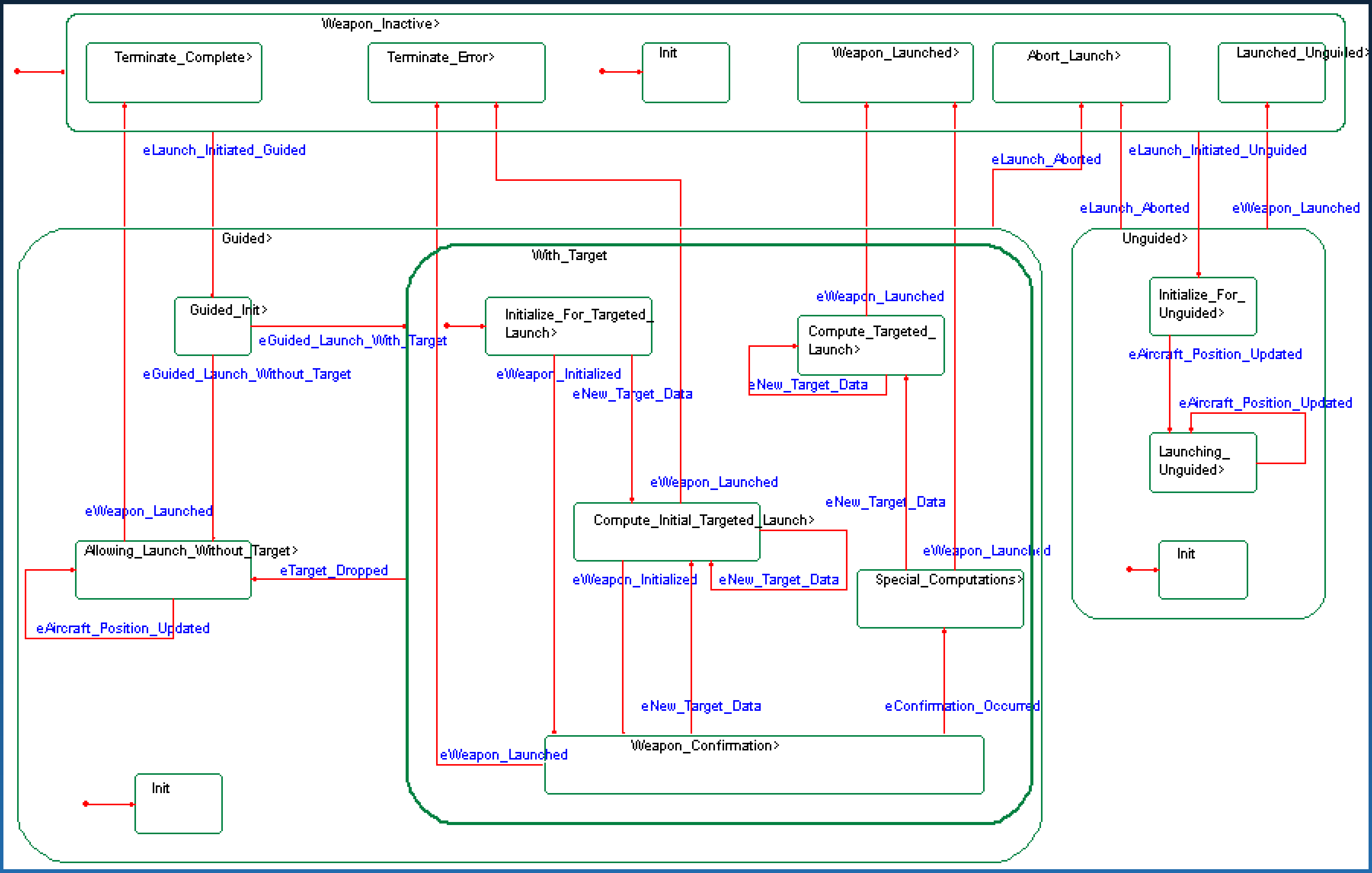
Problem Description - Application Internal Details



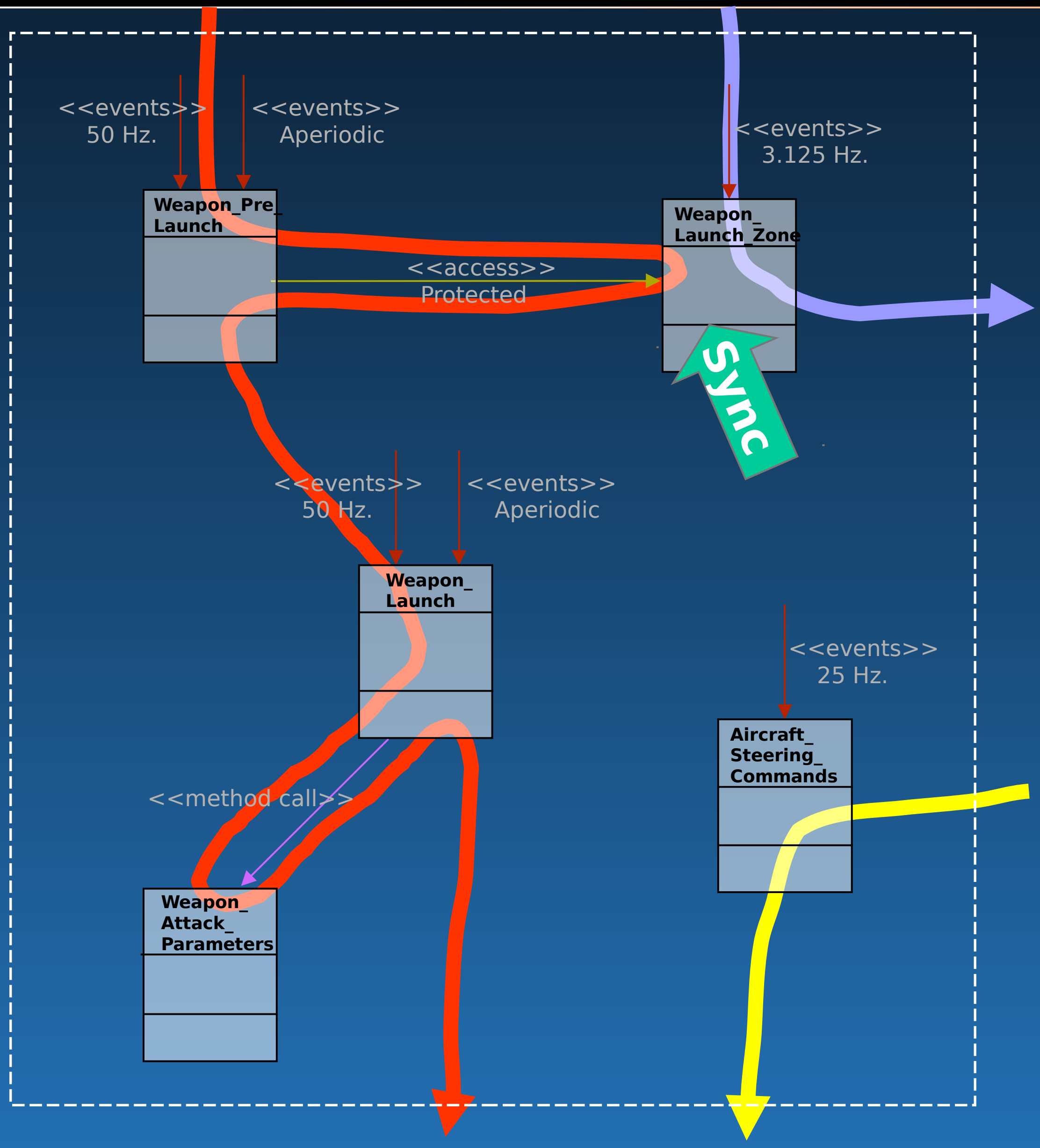
- **Capability package is a portion of a main program, residing on a single processor**
- **Designed to be added and removed as a unit**
- **Objects are selected within the package to address different aspects of the information processing**
- **Multiple system threads pass through this package**
- **Example shows required middleware services and various types of object interactions**
- **Detailed object finite state machines**
- **Pseudocode used to show interactions and middleware bindings**
- **Sensitive data elements (attributes) and computations have been removed**



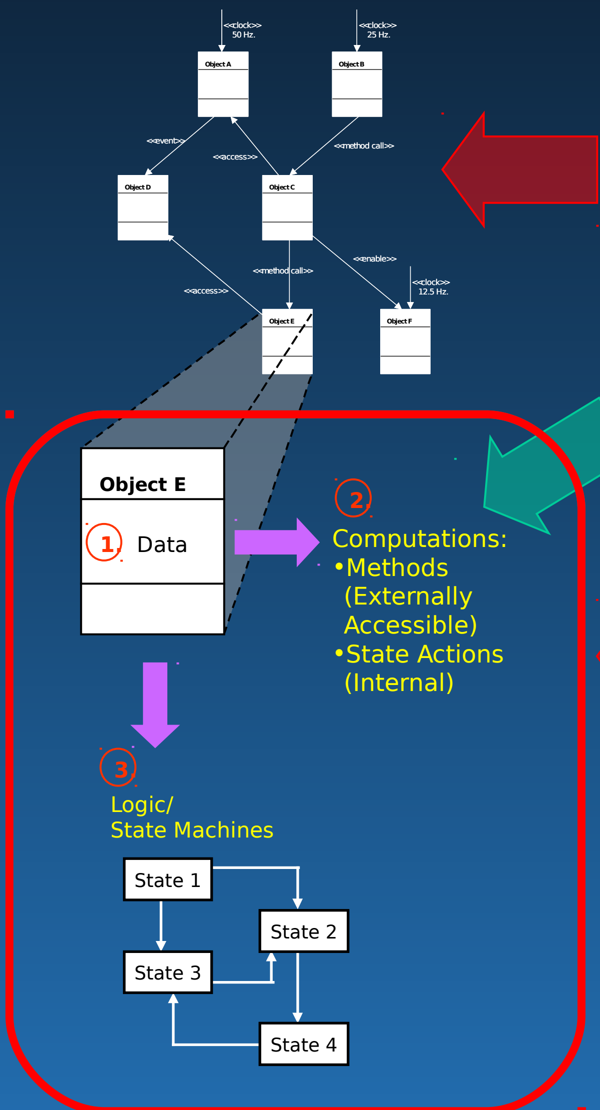
Problem Description – HFSM Details



Problem Description – System Threads



Problem Description - Scope of Current Tool Support



- **This is not a currently supported UML CASE view (object dependency diagram)**

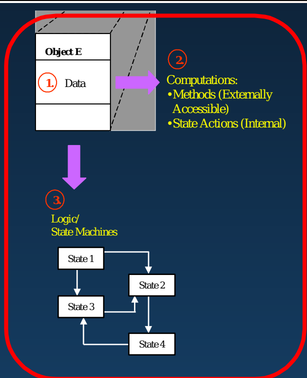
- **Content capture and OO structuring is well supported with current tools - SOLVED PROBLEM.**

Commercial OO CASE Capabilities END HERE.

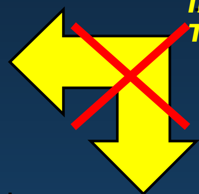
No current decision making support for:

- **threading**
- **distribution**
- **reconciliation of non-functional requirements and constraints:**
  - **timing, bandwidth, jitter, memory**
  - **reliability, fault tolerance**

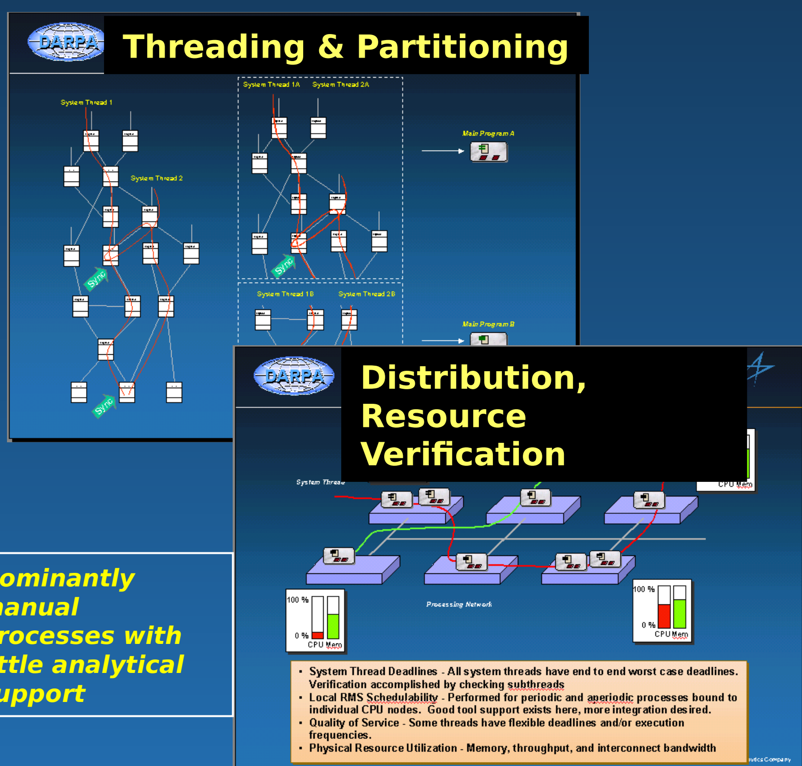
Problem Description - Summary



Content Capture



No Seamless Information Transfer



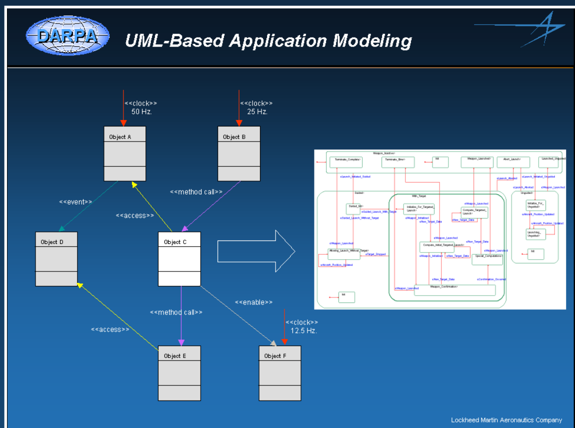
Dominantly manual processes with little analytical support

- **Non-functional requirements influence designs developed using commercial CASE tools. Early reconciliation of these issues is highly valuable**
- **We'd like an integrated CASE suite that supports the engineering activities required to transform captured content into a functioning embedded system**
  - **Non Functional Requirements Capture / Co-Representation**
  - **Design Trades/Analysis Support**
  - **Support for Verification of Non-Functional Requirements**

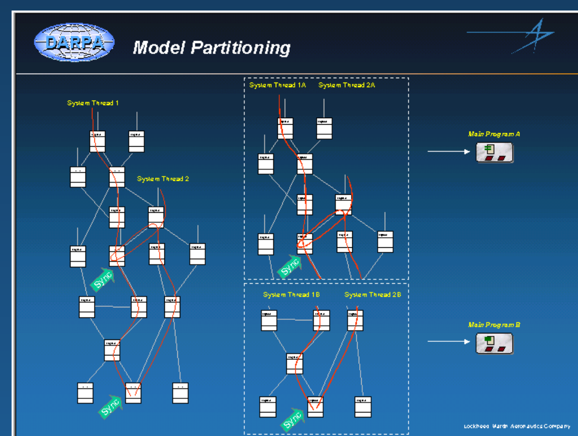
Problem Description – Practice Overview

- Different Tools
- Manual Information Transfer
- Inherent Cross Cutting Dependencies
- No Analytical Trades Support
- Inherent Feedback (Late)

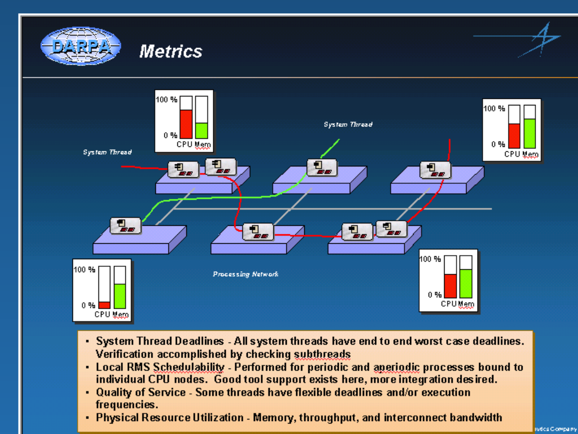
Content Capture



Threading & Partitioning



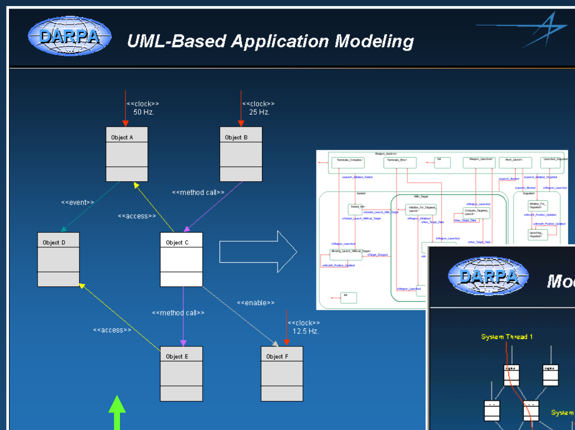
Distribution, Resource Verification



Problem Description – F-35 Use Case

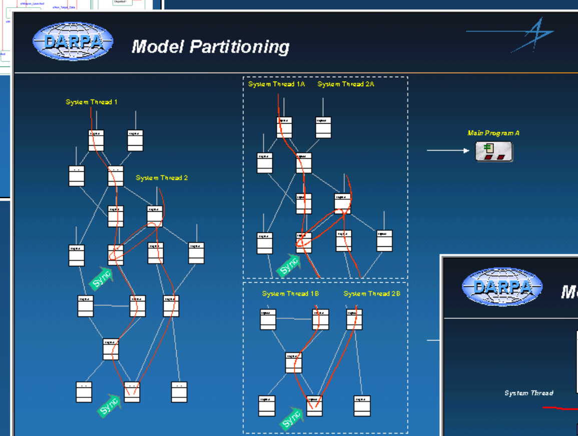
Content Capture

Replace Ad-Hoc Practices with Analytic Practices

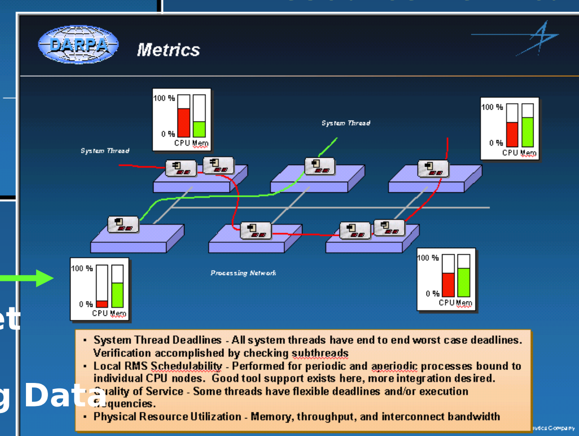


Threading & Partitioning

Reduce Scrap and Rework
Improve Overall Design Quality



Distribution,
Resource Verification



Record and Co-Represent Functional and Non-Functional Requirements

Execute Content Incrementally on Target
Animate and Debug at CASE Level
Capture Resource Utilization and Timing Data

Program Objective

- Provide a Complete Experimental Context for Engineering and Generating Avionics Applications using New Integrated CASE Technologies
- Evaluate MoBIES Technologies using Production Avionics Metrics and Historical Comparative Data from Major Weapon Systems Programs
- Demonstrate Benefits of Integrated, Multi-View CASE Technologies
- Transition Technology to Multiple Major Weapon System Programs

Program Objective – Execution

Approach:

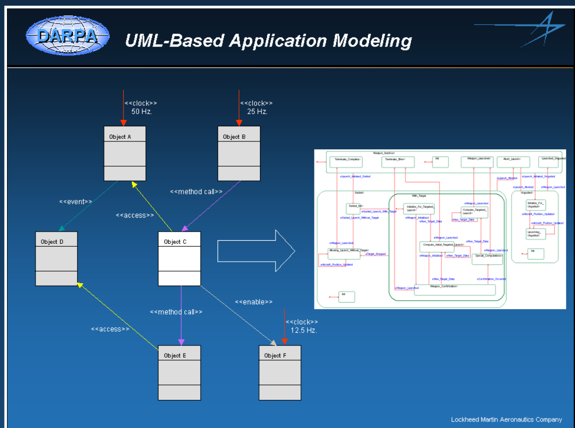
- **Allow Phase 1 / OEP experiments to solidify**
- **Perform supplementary experiments using "off the shelf" Phase 1 capabilities (I.e., no customization of Phase 1 products required for Lockheed Martin)**
- **Offer challenges / collaborations that are complimentary to the Boeing OEP**
  - **Intra-Component Capture and Utilization of Cross-Cutting Constraint Information**
  - **Formal methods, properties checking**
  - **Fault tolerance within multiprocessor**

Program Objective – Experiment 1

Assess Semantics of TimeWeaver Representation from the Perspective of the LM Challenge Problem:

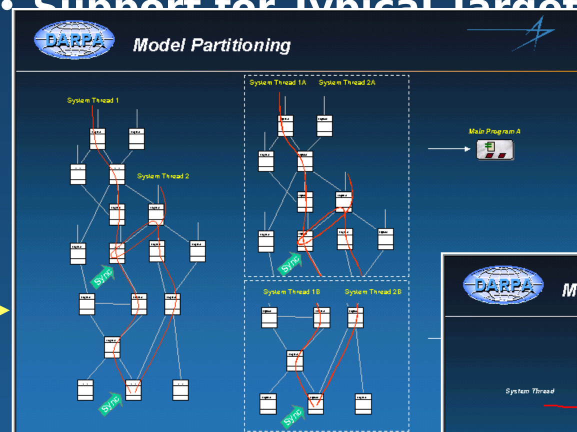
- **Scope - Component-Level and Intra-Component**
- **Compatibility with Primary CASE Tool Semantics**
- **Support for Specified Set of Avionics Constraints**
- **Support for Typical Target System Design**

Content Capture



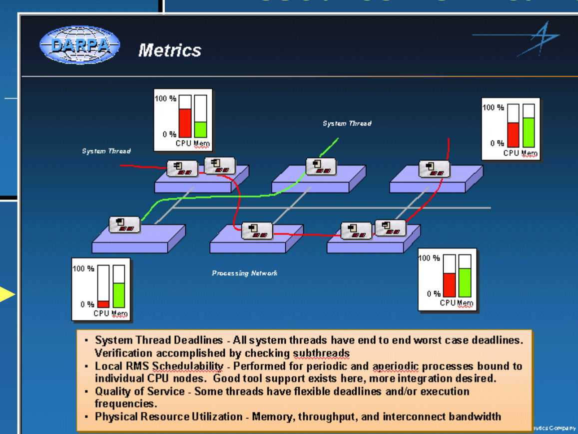
TimeWeaver

Threading & Partitioning



TimeWiz

Distribution, Resource Verification



Program Objective – Results Summary

Component Level Semantics

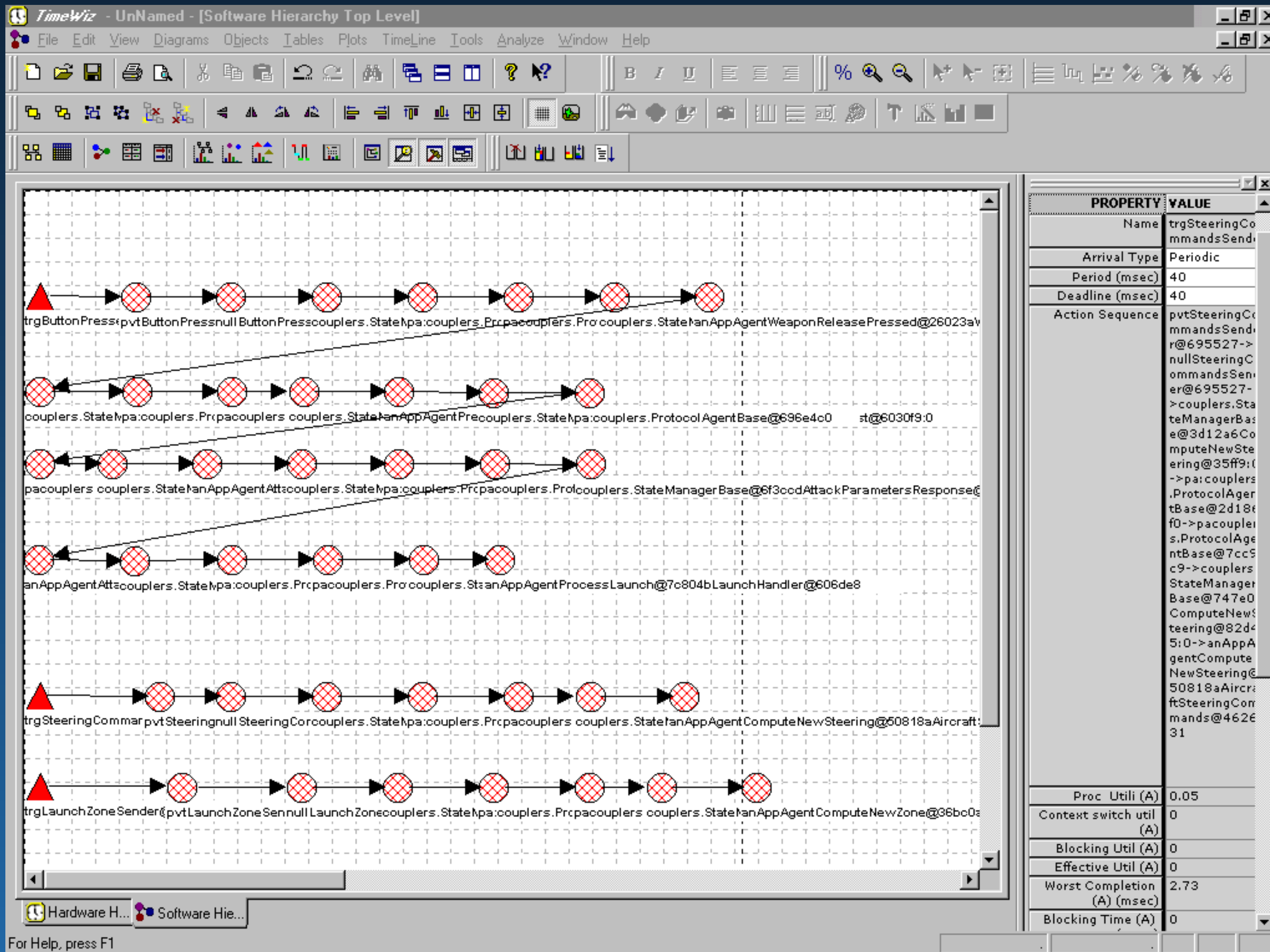
Definition of a Component	Meets +
Hierarchical Composition	Partial
Component Allocation in Multiprocessor	Meets
Multi-Threaded Components	Partial
Asynchronous Event Communication	Meets +
Asynchronous Data Communication	Meets +
Periodic and Event Driven Component Initiation	Meets +
Pre-Emptive, Priority Based Execution (CPU & Interconnect)	Meets
Computer Topology and Capacity	Meets
Abstraction of Physical Protocols	Meets
Representation of Threads Traversing Multiple CPU's	Meets
RMA, Deadline Checking, Bandwidth and Memory Analysis	Meets
Redistribution of Components (Trades)	Meets
Synchronization Protocols (Priority Inheritance)	Meets
Multicast Communication (Anonymous P/S)	Meets +

Program Objective – Results Summary

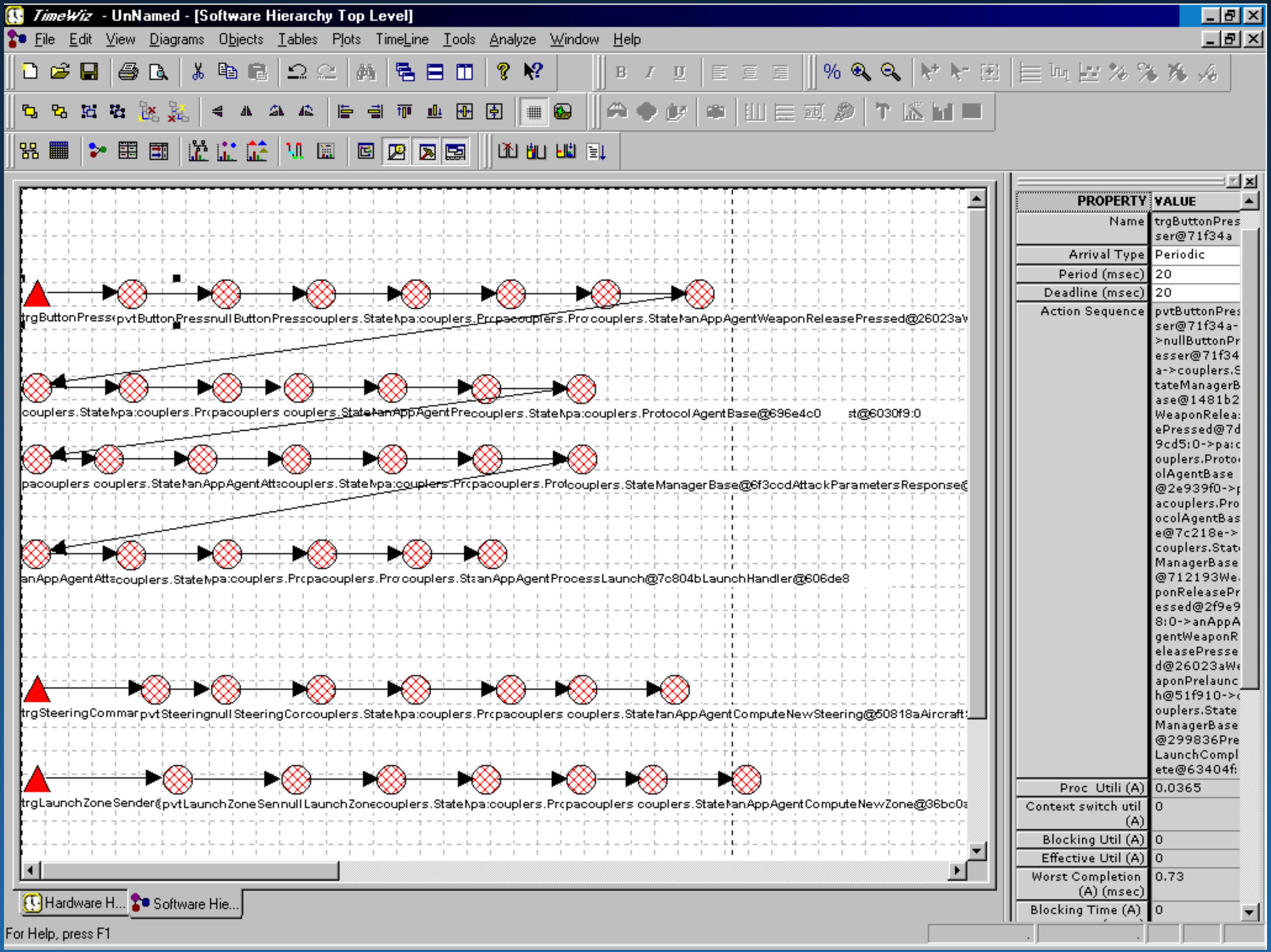
Object Level Semantics

Definition of an Object (Substituted Component) Meets +	
Hierarchical Composition	Partial
Active and Passive Objects	Meets
Method Invocation Semantics Needs Work	
Multiple Interacting Threads Partial	
Threads Traversing Multiple Objects	Meets
Asynchronous Interactions	Meets +
Abstraction of Interaction Protocols	Meets
Hierarchical Composition (Aggregation) Partial	
Locks & Mutexes	Meets
Interaction Directionality	Partial
Abstraction Features for Intra-Component Engineering Needs Work	
User Interface Assessment	Not Performed
Scalability	Not Performed

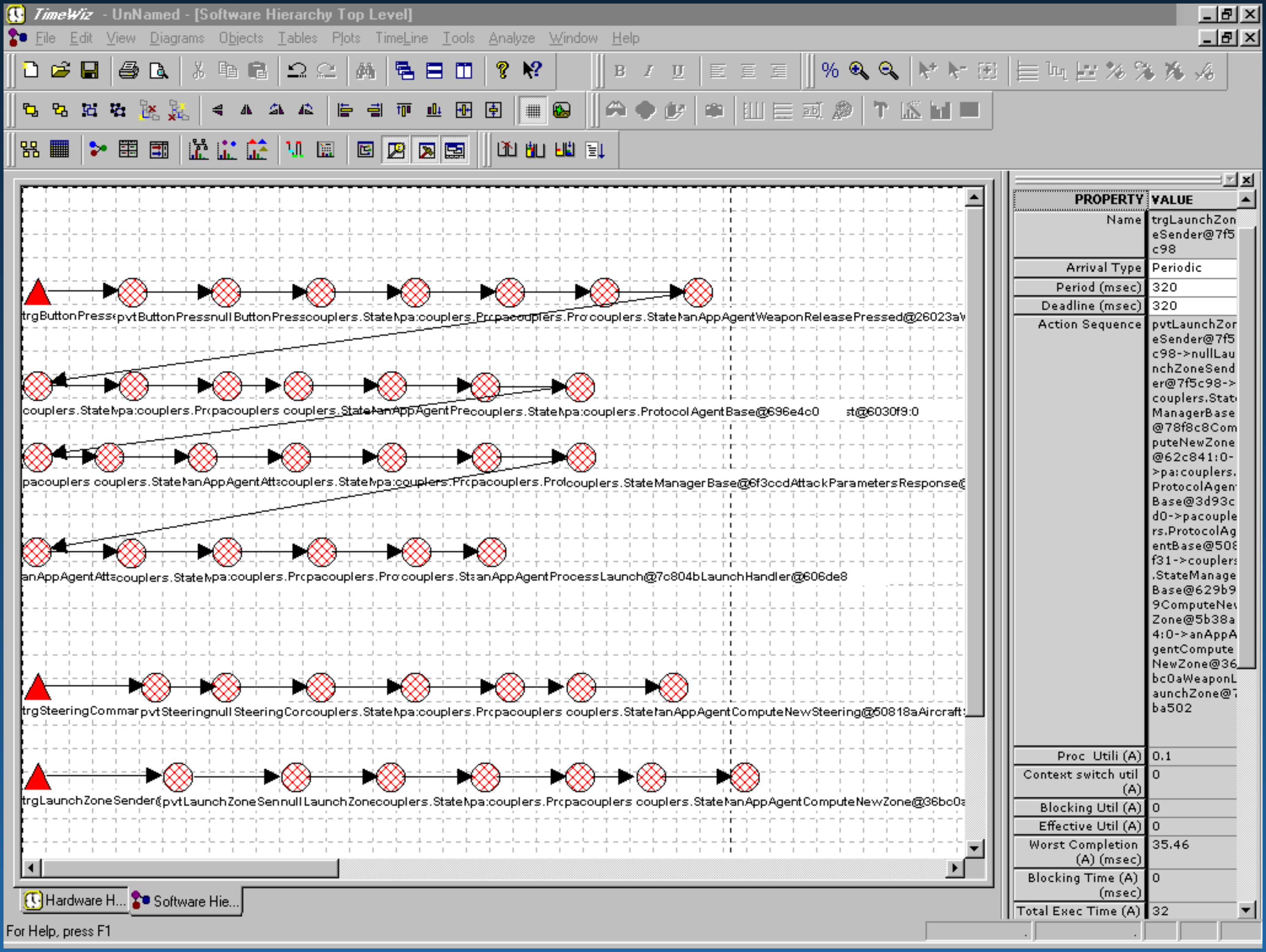
Tool Description – TimeWeaver / TimeWiz



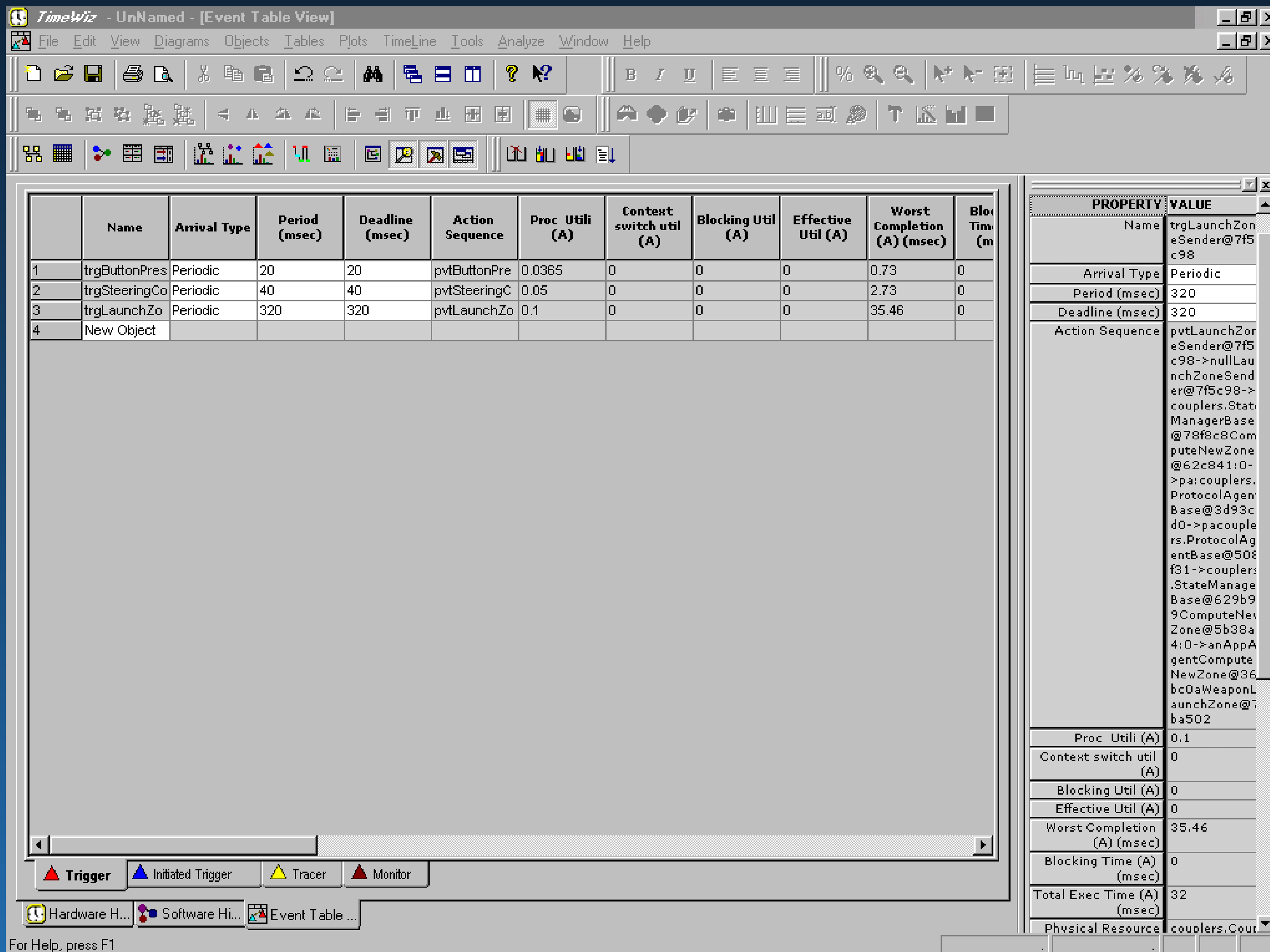
Tool Description – TimeWeaver / TimeWiz



Tool Description – TimeWeaver / TimeWiz



Tool Description – TimeWeaver / TimeWiz



OEP Participation

N/A

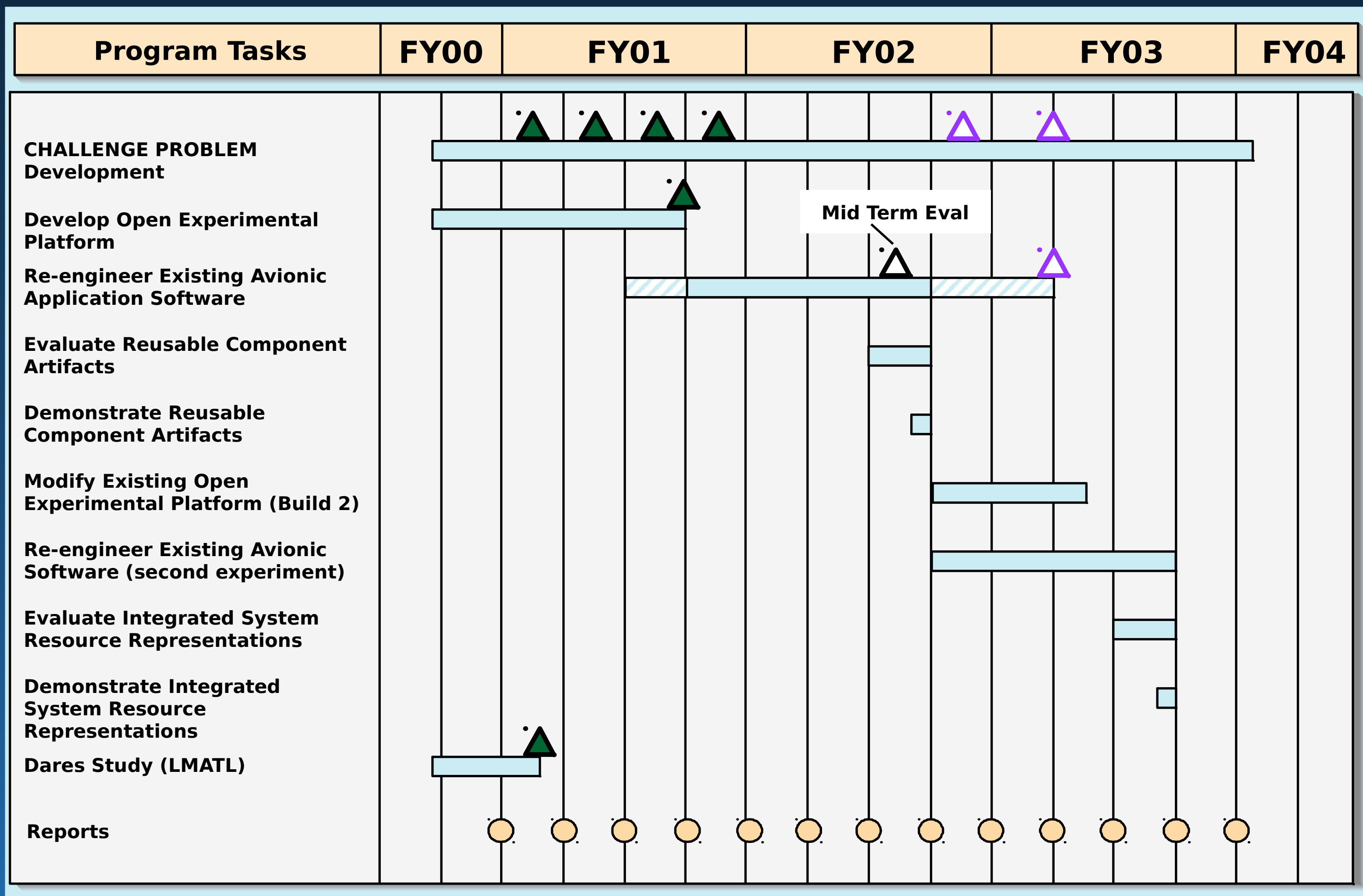
Project Status

- Challenge Problem White Paper is Being Updated (Version 5 in Work)
- Results from TimeWeaver Experiment Collected
- Weapon Release Example – Rhapsody Application in Development
- New Sample Application for Formal Methods and Test Vector Generation Experimentation being Assembled
- TimeWiz on F-35 List of Approved Tools. S/SEE Integration Copy is Currently Being Ordered

Project Plans

- Continue TimeWeaver Experiment, Provide "Full Integration" Demonstration
- Expand Collaborations with Phase 1 Teams in Areas of Formal Methods and Fault Tolerance
- Expand Challenge Problem Document (Version 6) to Include:
    - Avionics Fault Tolerance Application Specifics
    - Verification and Validation Challenges

Project Schedule and Milestones



Technology Transition/Transfer

LM Proven Path Commonality Initiative

- **Common Development Environments and Methods**
- **Common Architectures**
- **Cross Platform Reuse**
- **COTS Exploitation**

LM Advanced Avionics Architecture

2005

1998

Program Issues

- None

Backup Slides

Cruise Energy Management Function

SCR Logic Table

File

KeySelected Mode Transition Function

Name KeySelected

Source Mode	*Events*	*Destination Mode*
NoKey	@T(ButtonPressed) AND NumericKey'=True	FirstKey
FirstKey	@T(ButtonPressed) AND NumericKey'=True	SecondKey
SecondKey	@T(ButtonPressed) AND NumericKey'=True	ThirdKey
ThirdKey	@T(ButtonPressed) AND NumericKey'=True	FourthKey
FirstKey	@T(ButtonPressed) AND (ButtonSelected'=CLR OR ButtonSelected'=BACK) AND NOT NumericKey	NoKey
SecondKey	@T(ButtonPressed) AND ButtonSelected'=BACK	FirstKey
ThirdKey	@T(ButtonPressed) AND ButtonSelected'=BACK	SecondKey
FourthKey	@T(ButtonPressed) AND ButtonSelected'=BACK	ThirdKey
SecondKey, ThirdKey, FourthKey	@T(ButtonPressed) AND ButtonSelected'=CLR	NoKey

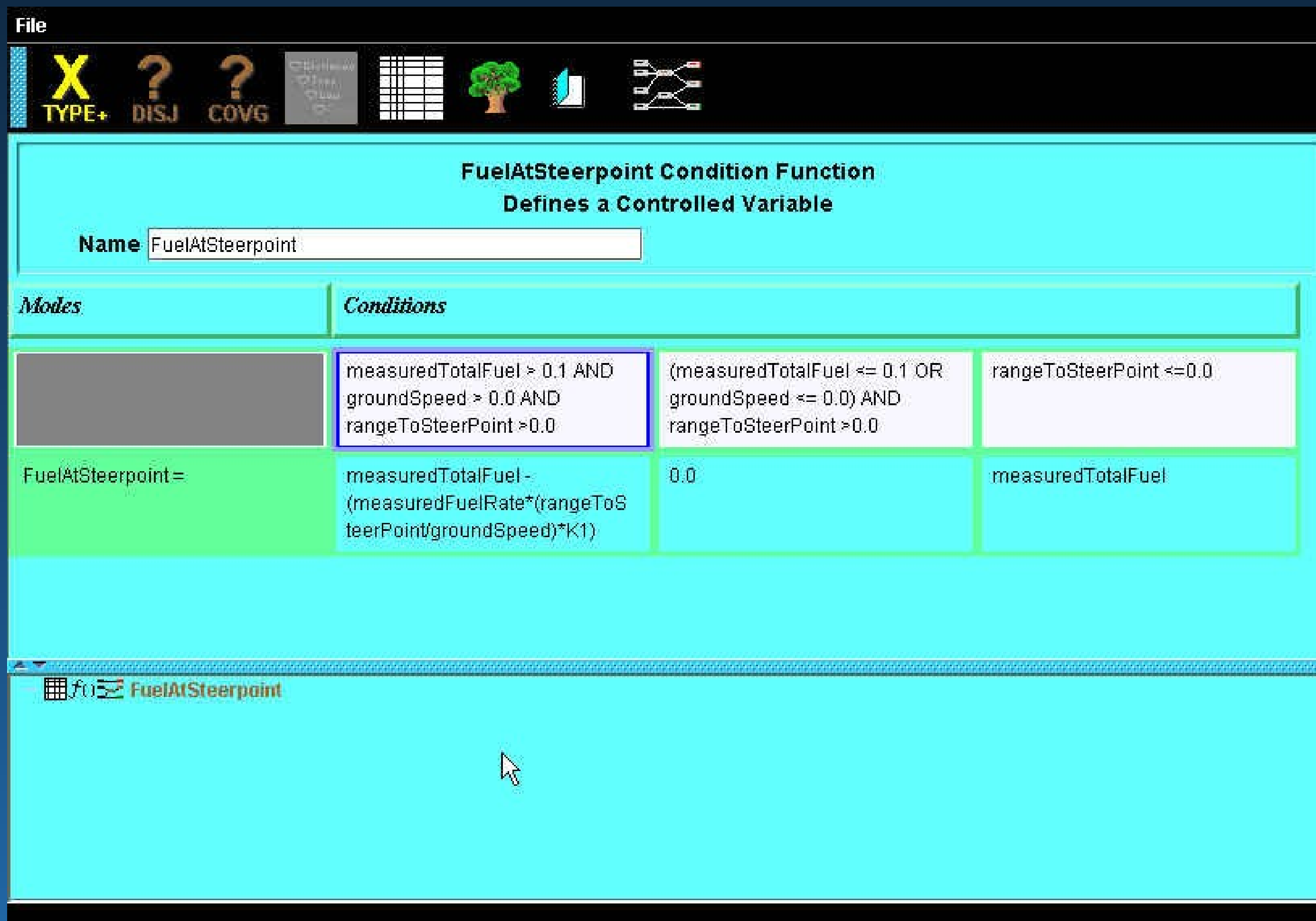
KeySelected

Cruise Energy Management Function

Optimal Mach Computation Equations

Cruise Energy Management Function

Fuel at Steerpoint Computation

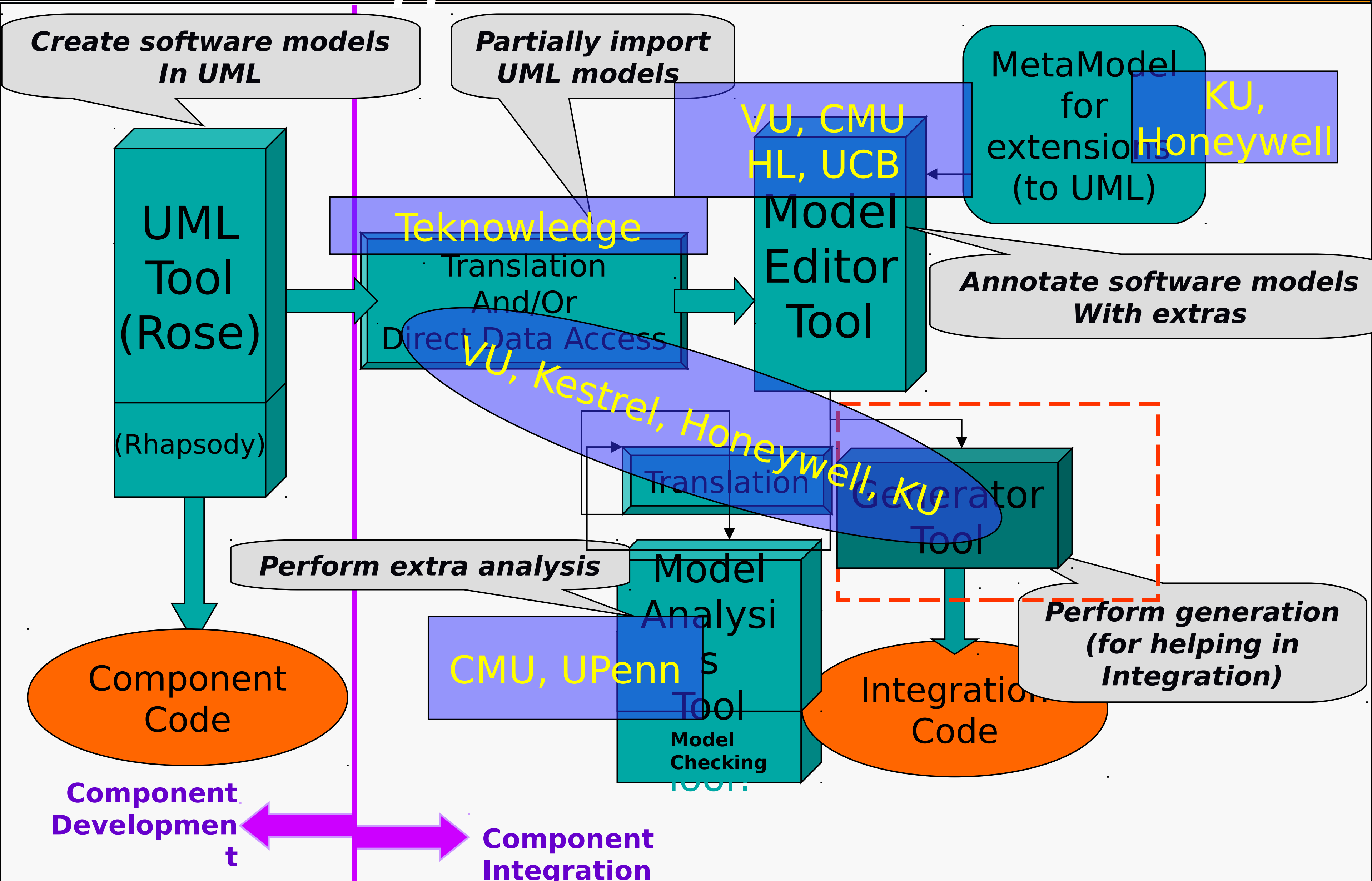


Cruise Energy Management Function

Automatically Generated Test Vectors

- FuelAtSteerpoint_vectors.html
- KeySelected_vectors.html

Weapon System OEP Phase I Integration Approach



Progress – Summary Since Last Meeting

- Challenge problem white paper has been updated with third segment on fault tolerance and is available for distribution
- Models and code from the weapon delivery example are being converted into Rational Rose form to be compatible with Boeing OEP (complete)
  - CORBA RT mappings are known
  - Rhapsody versions (with Harel state models) of identical functionality are available for experiments involving formal methods & model checking
- Mid Term Experiment planning activities in progress. Collaboration with CMU solidified.
- Modifications to the portable demonstration environment have begun

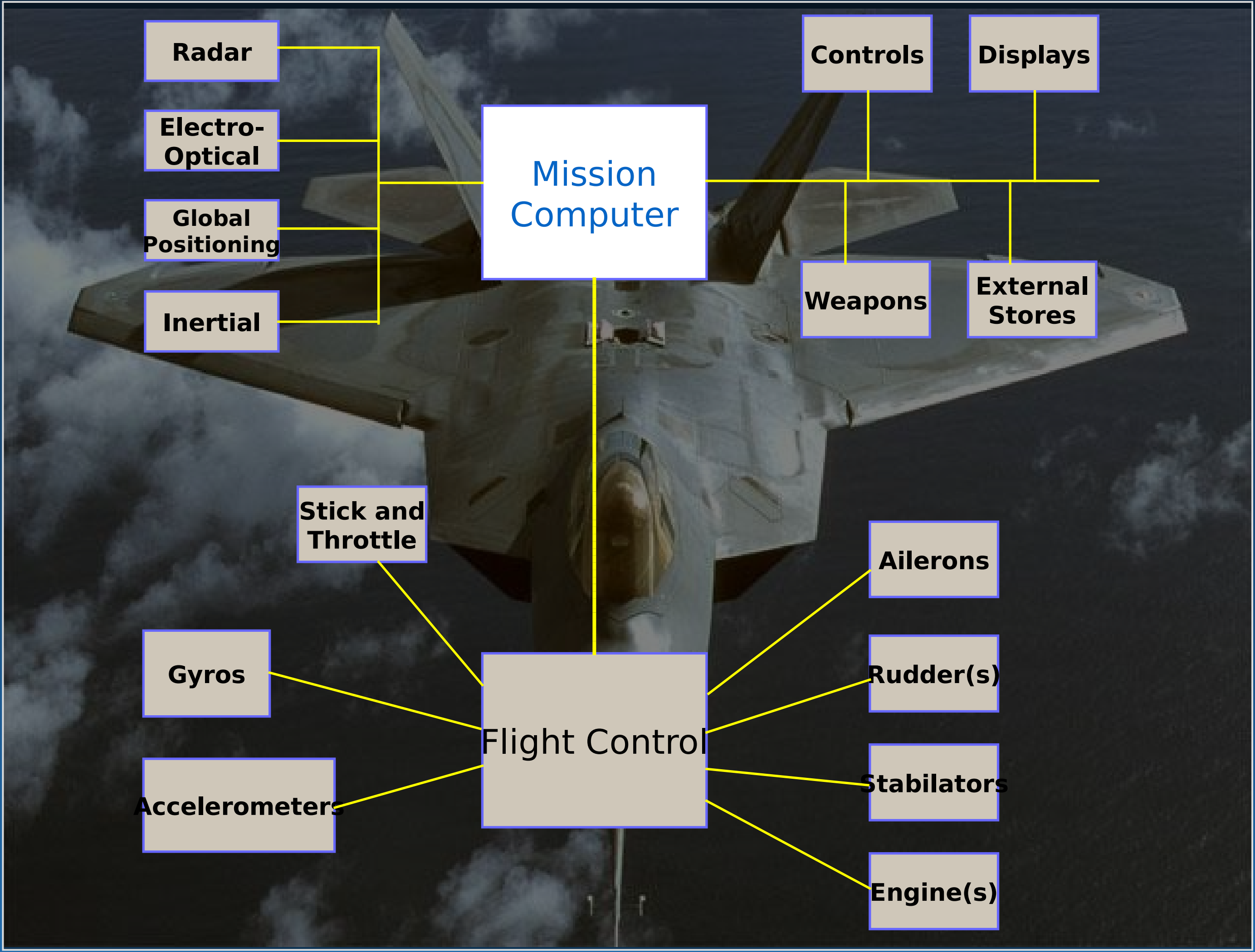
Introduction

- First draft generic characterization to be used as a starting point for Phase 1 and 2 collaborations
- Focuses primarily on multi-view CASE modeling challenges
  - Notations, views, semantics, design styles
  - Metrics, constraints, analyses
- “Common denominator” description distilled from multiple weapon system programs (F-35, F-22, etc.)
- Covers typical mission functions like situation assessment, fire control, navigation, tactical decision aiding, payload management
- Depicts modeling sequence, elaboration, application and reconciliation of constraints
- Experiments will use applications that have characteristics defined in this characterization

Assumptions & Caveats

- Mainstream commercial concepts and terms are used to the greatest extent possible
- Description draws heavily on UML, POSIX, Object Orientation, and real-time design practices
- Application focus:
  - Mission critical applications. Issues related to classification, security, and flight criticality have been deferred
  - No specialty applications (I.e., signal processing, closed loop control, display symbology generation)
- Scope tailored toward high value modeling improvements. Certain less critical details and issues have been deferred (security, cluster startup, etc.)
- Future versions will address V&V needs including topics of test vector generation and model checking

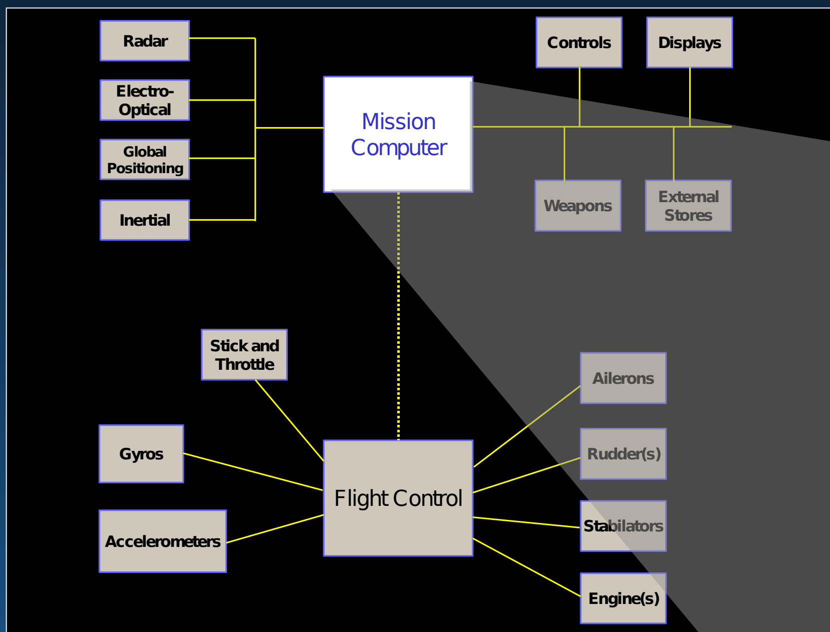
Challenge Problems - Context

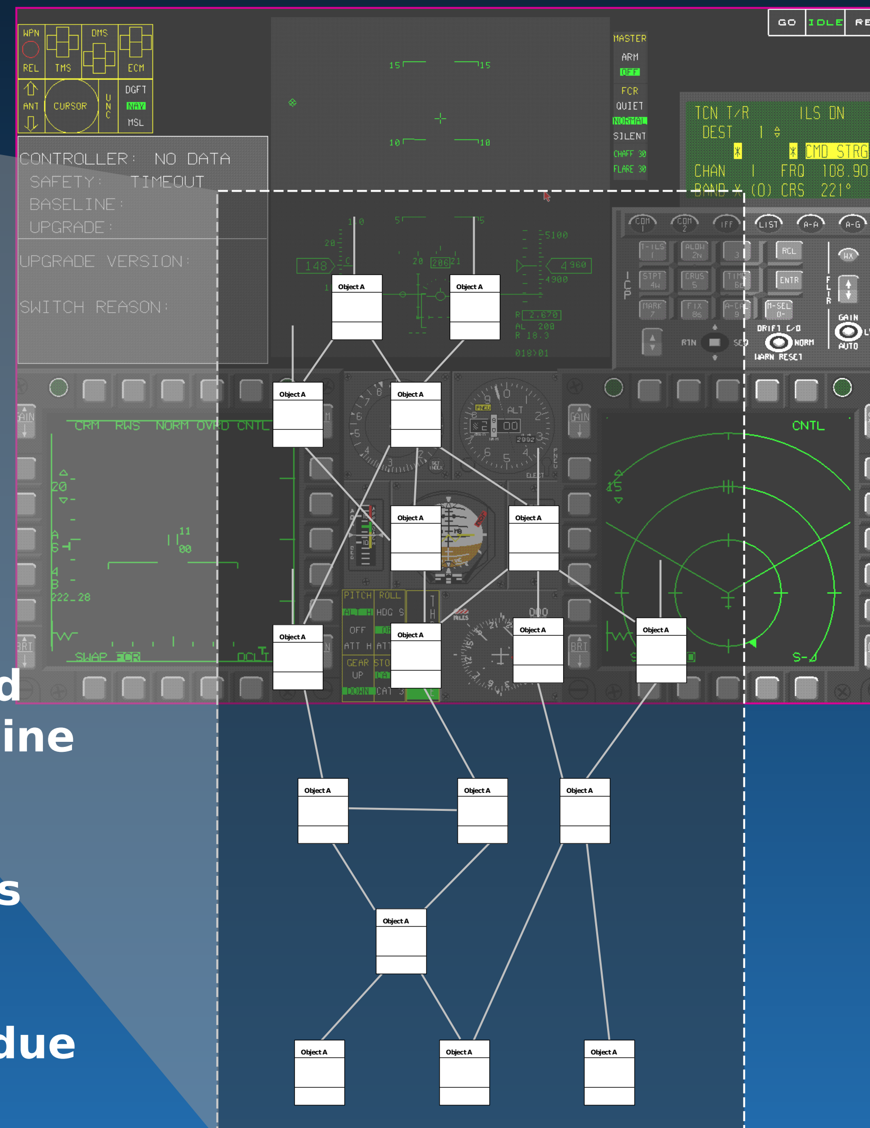


<u>Characteristics</u>

- Complex, Large
- Decades Lifespan
- Frequent Software Updates
- Mix of Computation Types
    - Logic/State Machine
    - Computational
    - Signal Processing
    - Feedback Control

Logical View



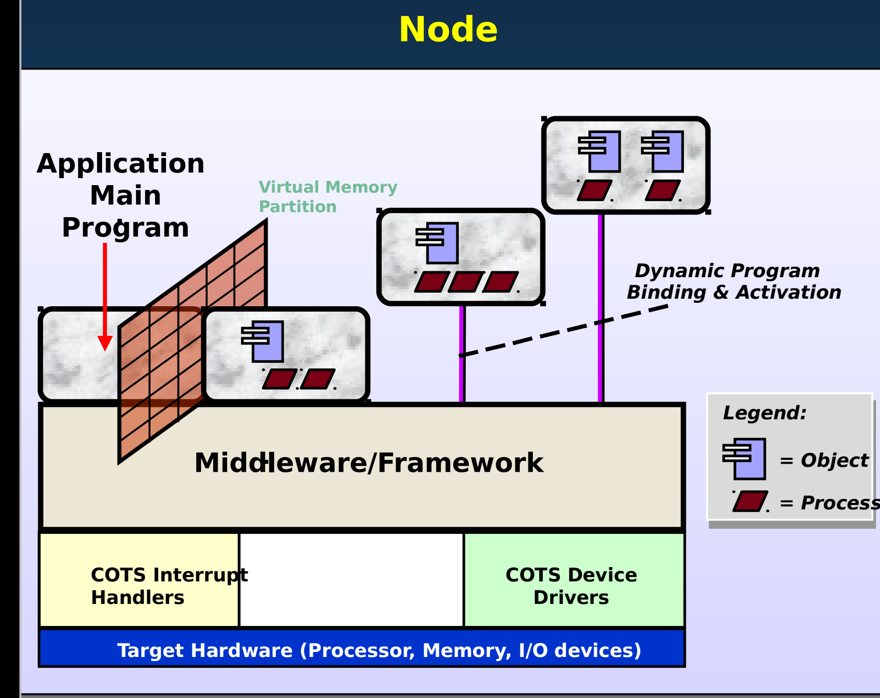


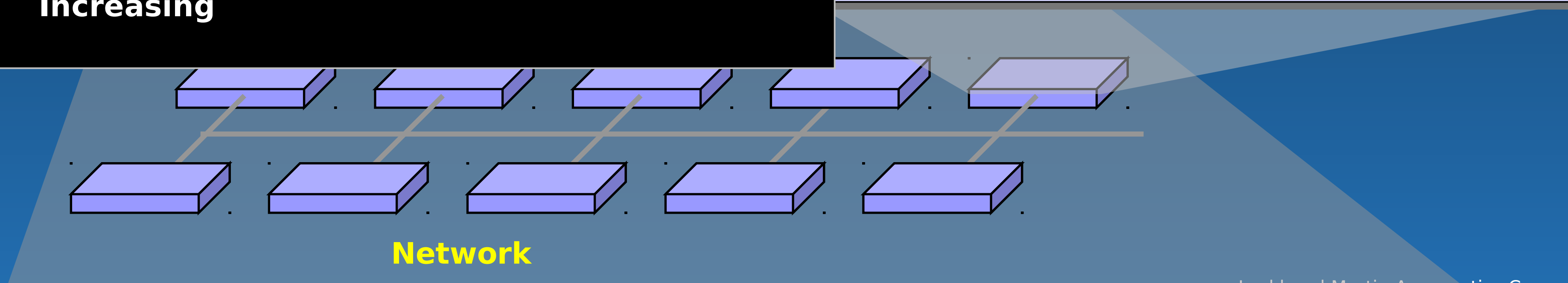
- Application “content” is dominated by: computation, finite state machine (FSM) and combinatorial logic
- Hundreds of objects / instances
- Cohesive - objects are chosen so as to minimize dependencies and interactions
- Object view is primary CASE view due to content variety and size
- UML notation is very useful - seeming value in creating new views

Physical View

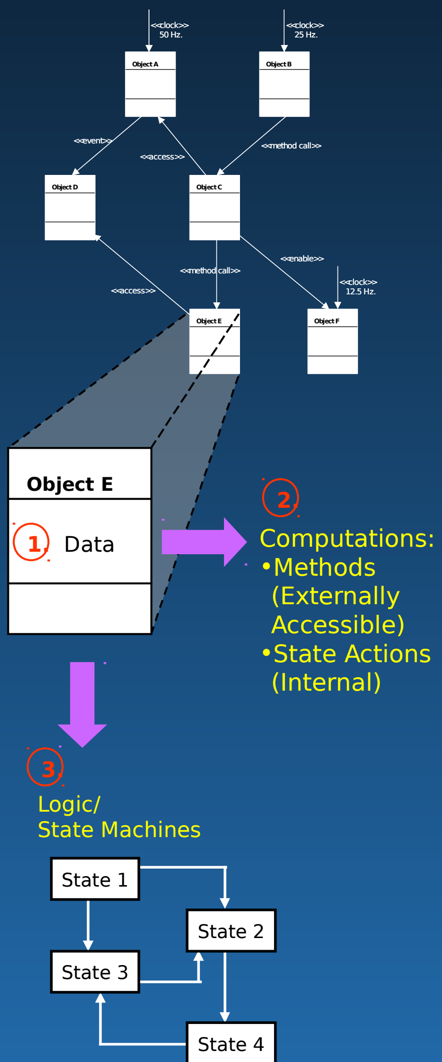
Design Characteristics:

- Shared Processing Network of General Purpose Commercial CPU's
- Computer Contains Some Specialized Processing and I/O Elements
- Multiple Collaborating Main Programs (both Synchronous and Asynchronous Message Passing)
- Each Main Program Contains Multiple Objects
- Multiple Main Programs Occupy Each Processing Node
- Dynamic Application Binding with Allocation Constraints
- System-Level Reconfiguration Requirements
- Both Fixed and Dynamic Processing Loads
- Both Hard and Soft Real-Time Requirements
- Total Application Sizes are 2 MSLOC and Increasing





Anatomy of a Typical Object



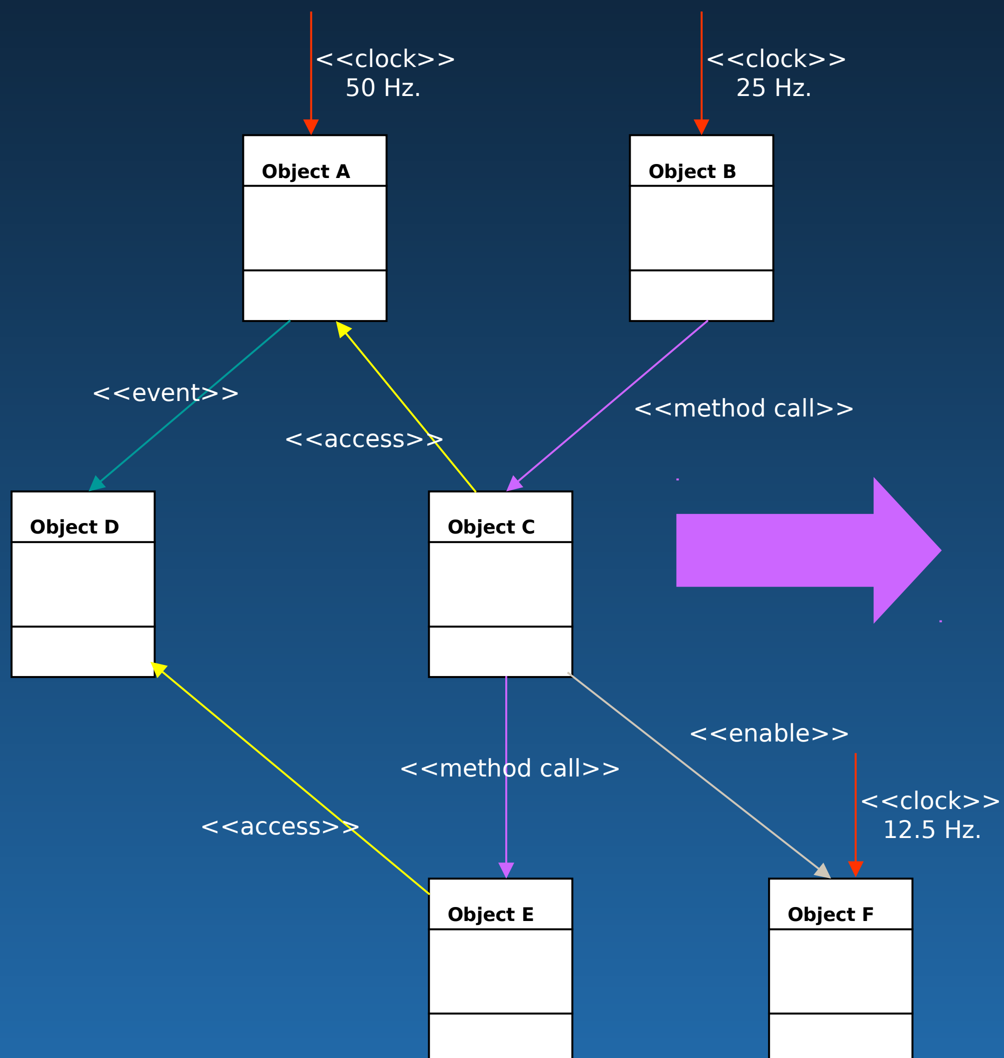
- Objects contain encapsulated data, accessor methods are generally provided
- An object may contain a finite state machine or combinatorial logic.
- FSM's generally have between 5 and 20 states
- An object may contain one or more computational procedures accessible as external methods
- Computations range in size from 10 to 5K lines of code
- Internal algorithmic procedures are often defined as responses to FSM transitions
- Matlab generated code would generally be integrated as an object method
- For MoBIES purposes, assume all generated and manually developed

Design Time Environment

Phase 1 Researchers can Assume:

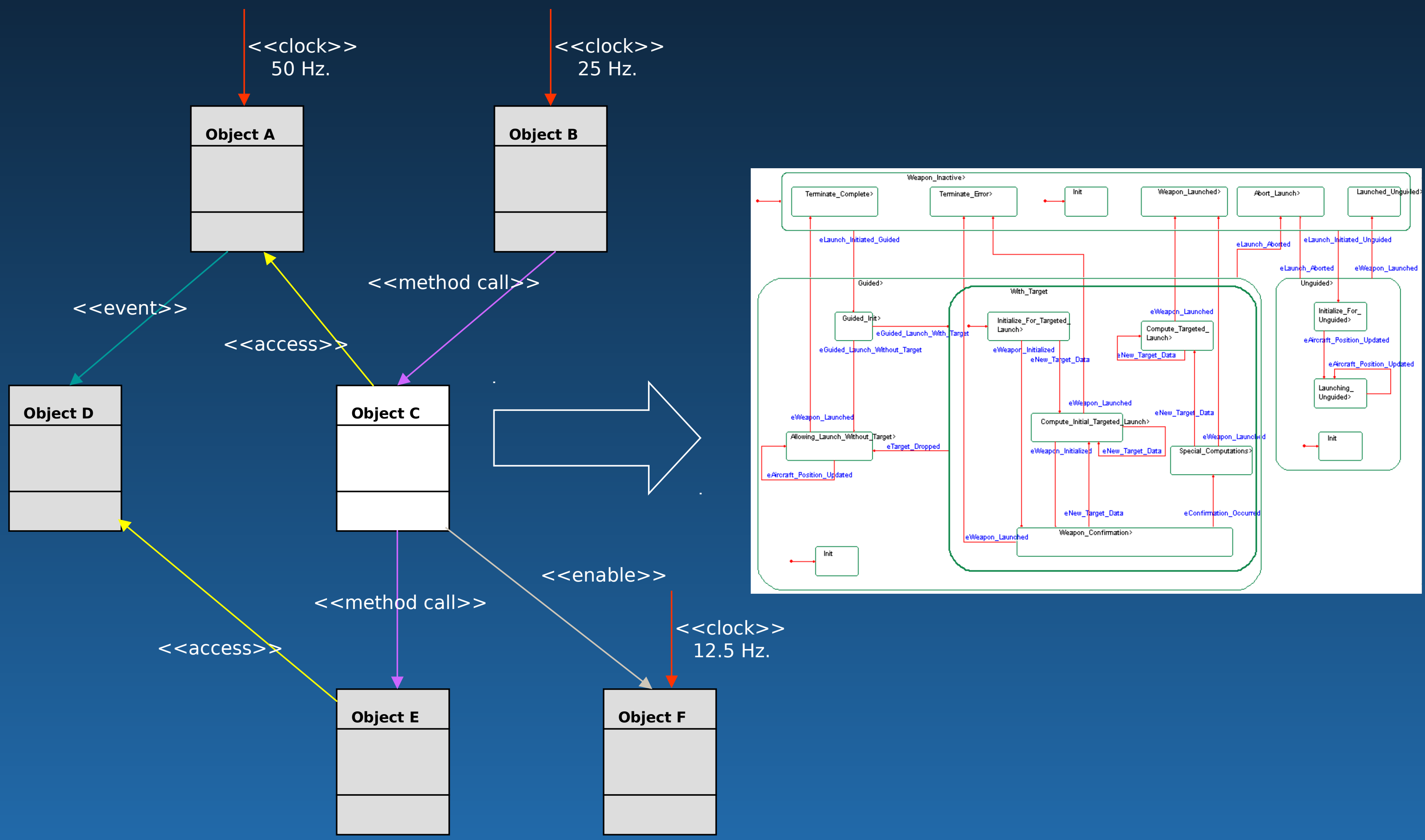
- **UML based CASE tool that supports object view and embedded FSM’s (Rhapsody, I-UML, Rose + Stateflow possible, not preferred)**
  - **Full code generation in C++**
  - **Harel semantics for FSM’s preferred, flat acceptable**
- **Algorithmic code from Matlab / Matrix X can be incorporated into the UML models as object methods or state actions, otherwise it is manually generated**
- **Target Framework Semantics (POSIX works, RT CORBA fine also)**
  - **periodic scheduling events, multitasking support, semaphores**
  - **prioritized asynchronous messaging (cross processor)**
  - **event combinatorial service (optional)**
  - **virtual memory, time and space partitions (optional)**
  - **Configurable binding capability (select semantically**

“Logical” Object Interactions

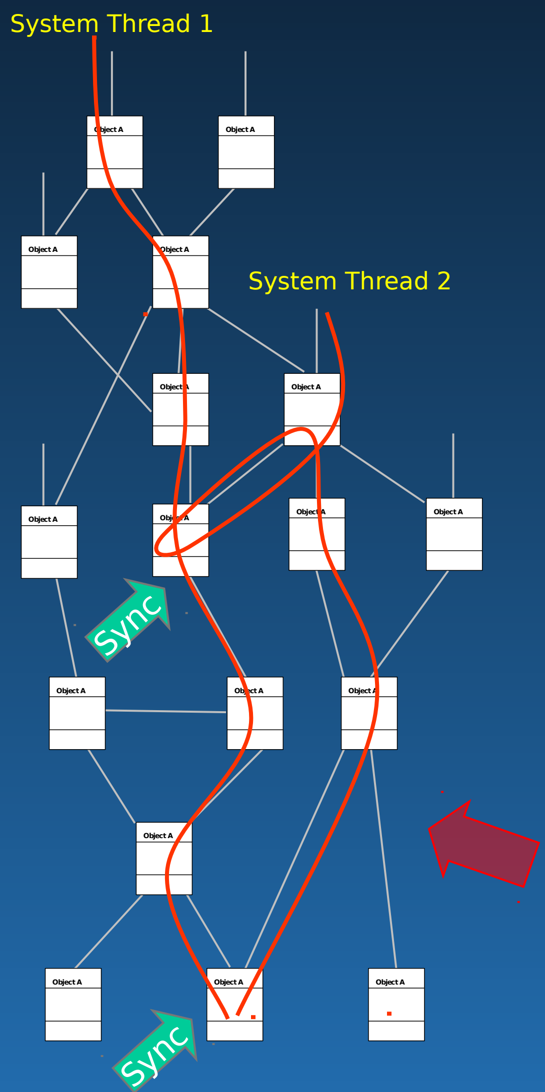


- **Five logical types:**
  - **FSM event,**
  - **Enable,**
  - **Access,**
  - **Clock,**
  - **Method call**
- **In CASE environment, all interactions mechanisms are procedure calls, except clock**
- **As models are evolved to the target platform, call substitutions (O/S, middleware, framework) are made for object interactions as needed**
  - **Cross processor events**
  - **Interprocessor messages, etc.**
- **Substitutions can change model semantics - designers need to keep track of this effect**

UML-Based Application Modeling



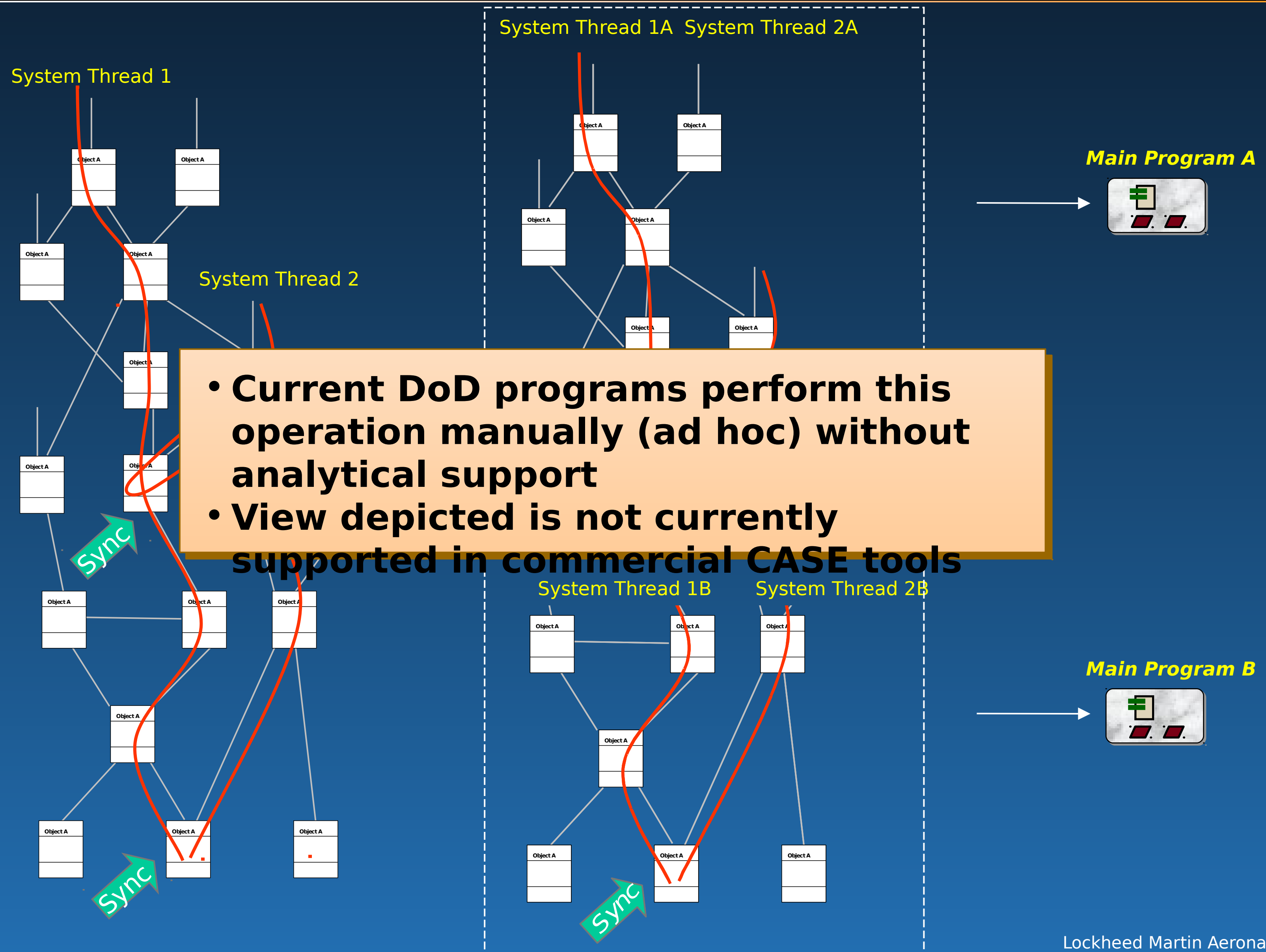
System Threads



- Majority are periodic and scheduled at harmonic rates.
- Some are aperiodic
- Number of system threads can range from 10 to 30.
- Most system threads are also mode dependent
- Synchronization points must be added within models to account for interactions
- Generally, forks and joins within a thread are rare
- System threads are mapped to heavyweight processes on the target physical platform

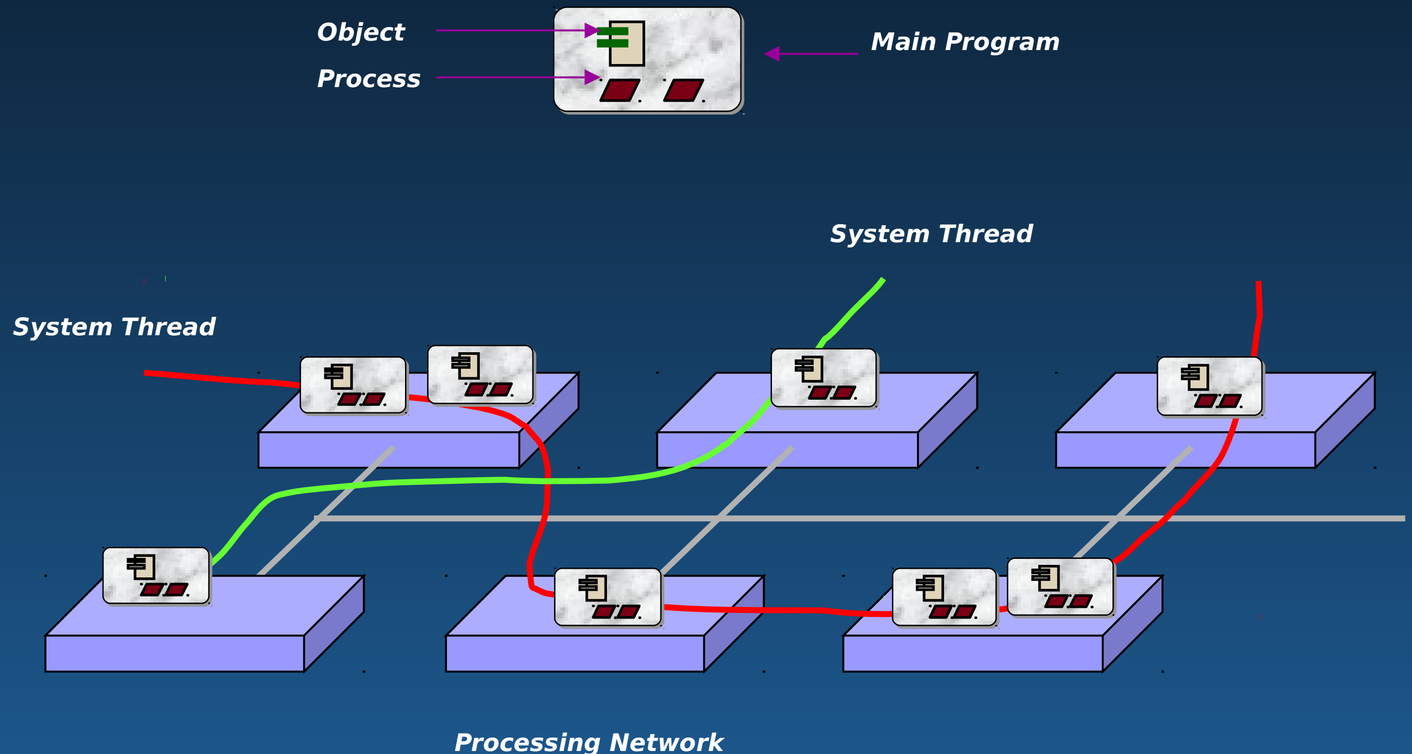
- **Current DoD programs perform this operation manually (ad hoc) without analytical support**
- **View depicted is not currently supported in commercial CASE tools**

Model Partitioning



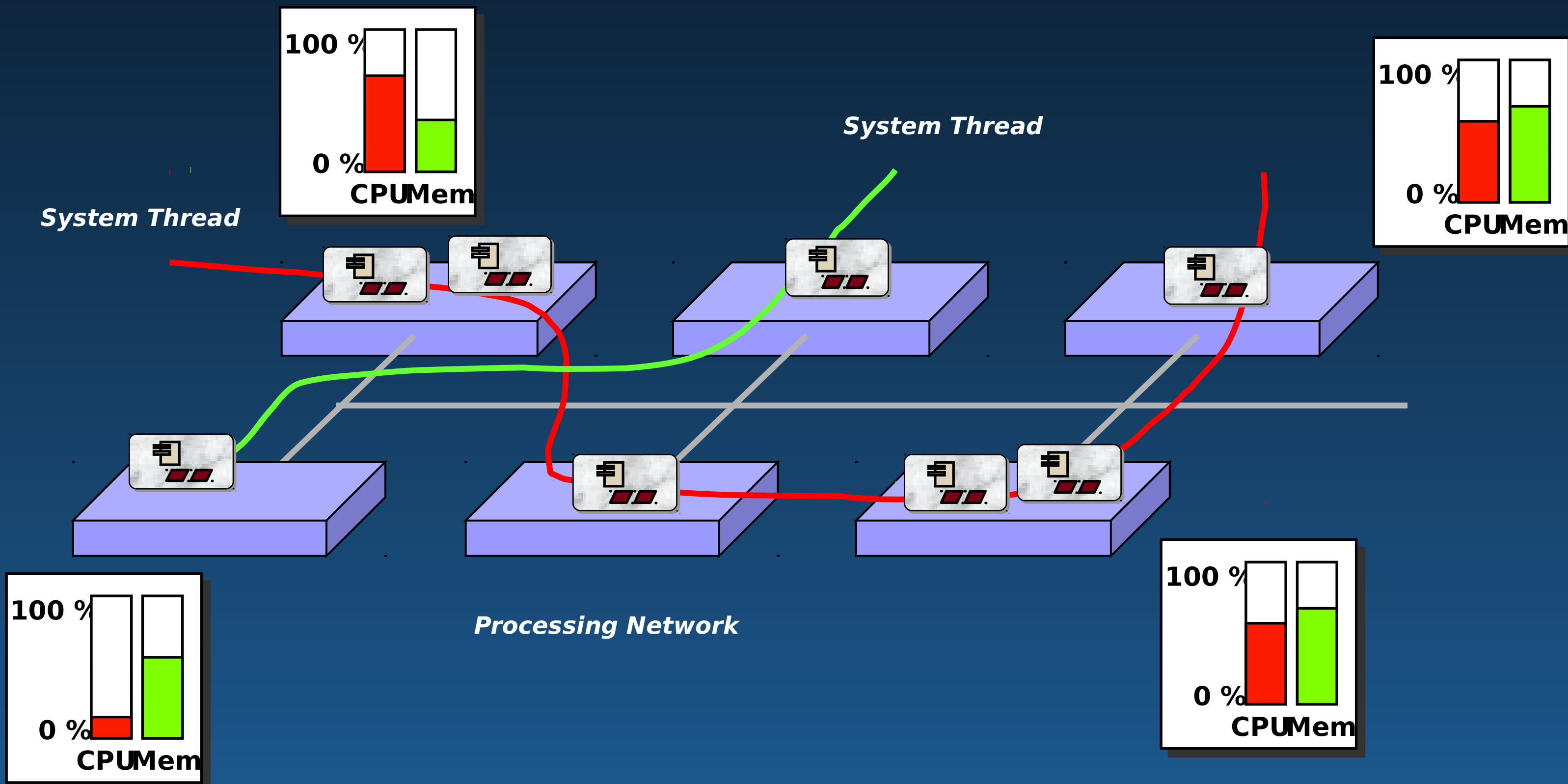
- **Current DoD programs perform this operation manually (ad hoc) without analytical support**
- **View depicted is not currently supported in commercial CASE tools**

Physical Distribution



- Current DoD programs perform this operation manually (ad hoc) without analytical support
- View depicted is not currently supported in commercial CASE tools

Non Functional Requirements - Metrics



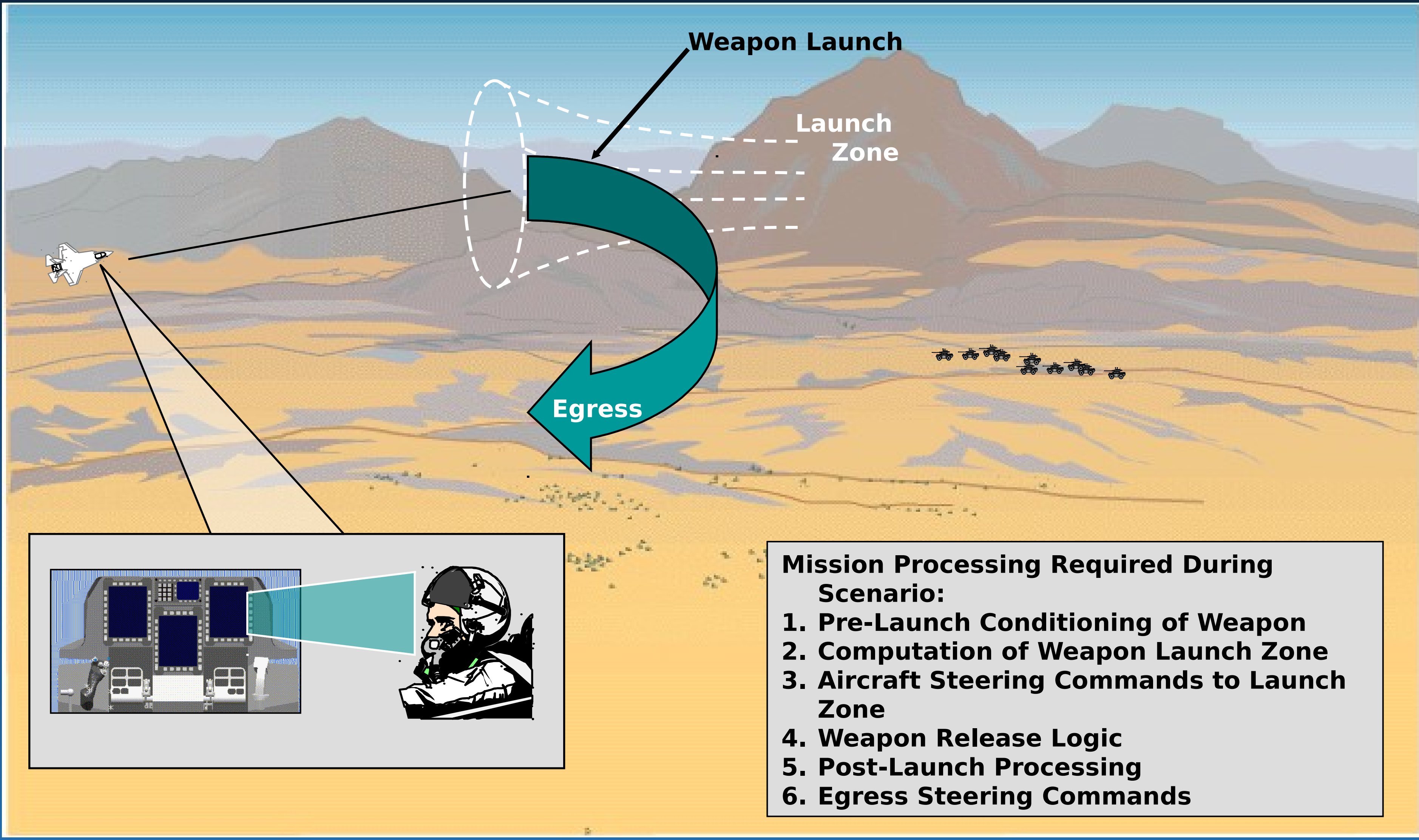
- System Thread Deadlines - All system threads have end to end worst case deadlines. Verification accomplished by checking subthreads
- Local RMS Schedulability - Performed for periodic and aperiodic processes bound to individual CPU nodes. Good tool support exists here, more integration desired.
- Quality of Service - Some threads have flexible deadlines and/or execution frequencies.
- Physical Resource Utilization - Memory, throughput, and

High Payoff Modeling Capabilities

Integrated CASE "workbench" supporting:

- **Key Views: 1. Object interaction, 2. Threading with temporal annotations, 3. Partitioning, Allocation, Distribution, 4. Resource Utilization and Schedulability.**
- **Integrated Analysis and Design Trades (temporal, resource, distribution). Rapid, early, accurate trades are seriously needed.**
- **High level thread simulation on virtual target.**
- **CASE level symbolic debug on target.**
- **Instrumentation - Ability to execute models or partial models on the target while capturing resource utilization and timing data that is then appended to integrated CASE representations.**

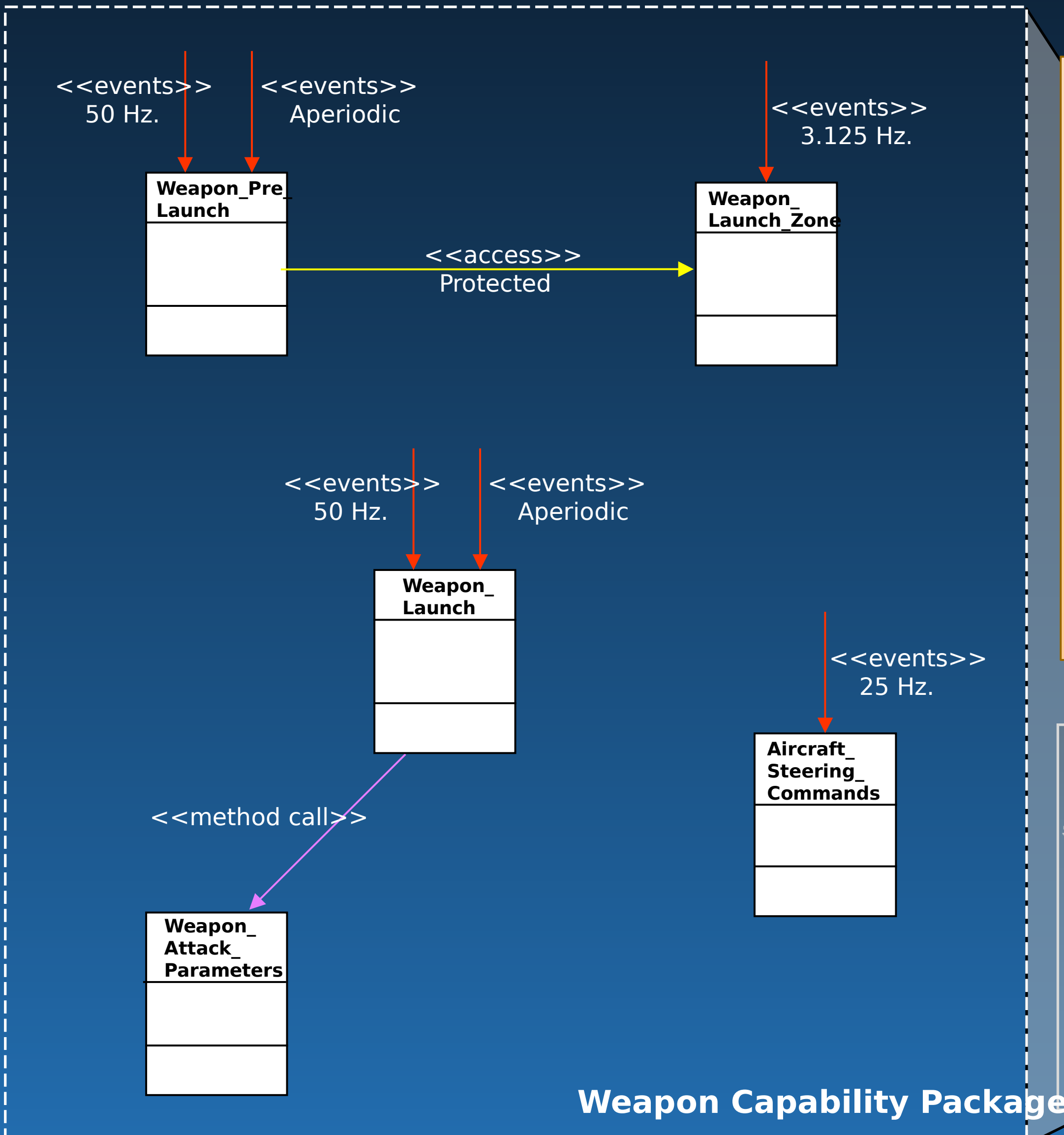
Example – Operational Scenario



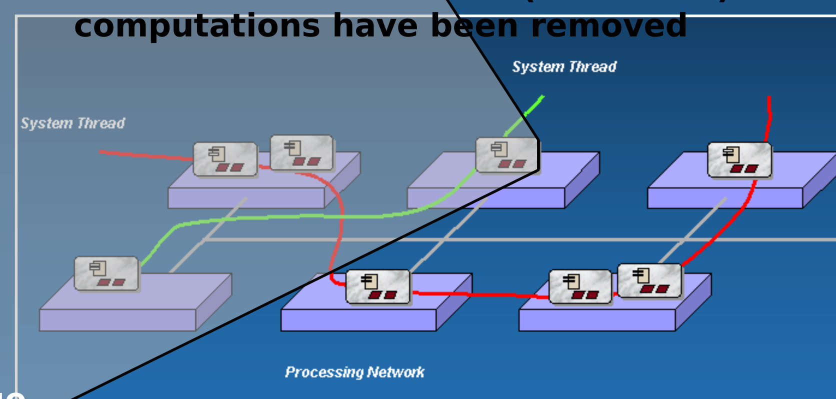
Mission Processing Required During Scenario:

1. Pre-Launch Conditioning of Weapon
2. Computation of Weapon Launch Zone
3. Aircraft Steering Commands to Launch Zone
4. Weapon Release Logic
5. Post-Launch Processing
6. Egress Steering Commands

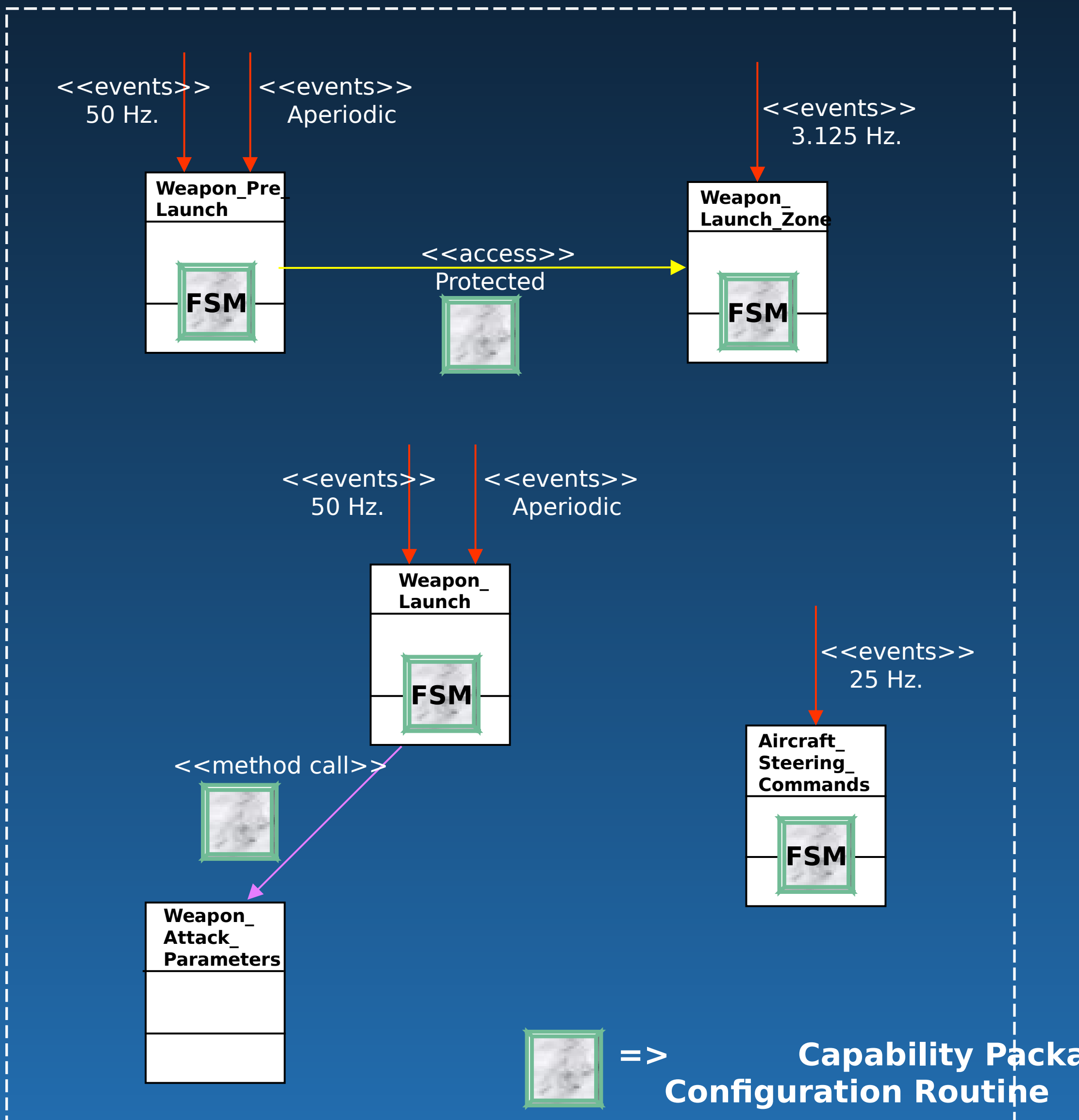
Application Example



- **Capability package is a portion of a main program, residing on a single processor**
- **Designed to be added and removed as a unit**
- **Objects are selected within the package to address different aspects of the information processing**
- **Multiple system threads pass through this package**
- **Example shows required framework services and various types of object interactions**
- **Detailed object state models**
- **Pseudocode used to show interactions and framework bindings**
- **Sensitive data elements (attributes) and computations have been removed**



Example – Main Menu



=> System Threads

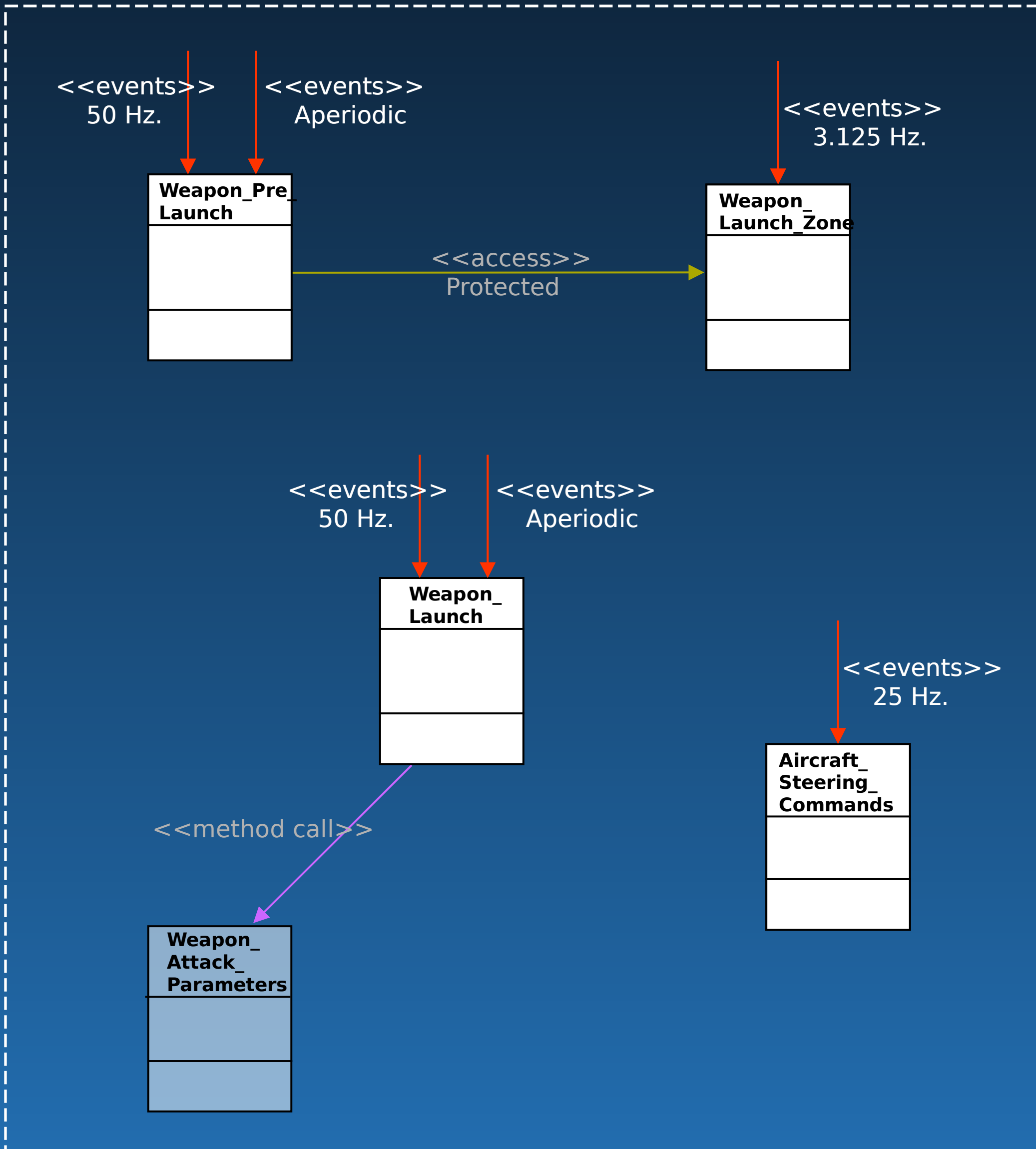
=> Event Signups

=> Next Presentation Segment

Other attributes of example application:

- **Represents about 12KSLOC, which is ~1% of the total size of a typical mission computer**
- **Near equal mix of computations and decision logic**
- **All functionality within this package is time critical**
- **Local threads are segments of cross-processor system threads**
- **Use of military specific I/O channels – this serves as a distribution constraint.**
- **Execution rates driven by a number of factors including capacity limitations, accuracy requirements, and human factors**

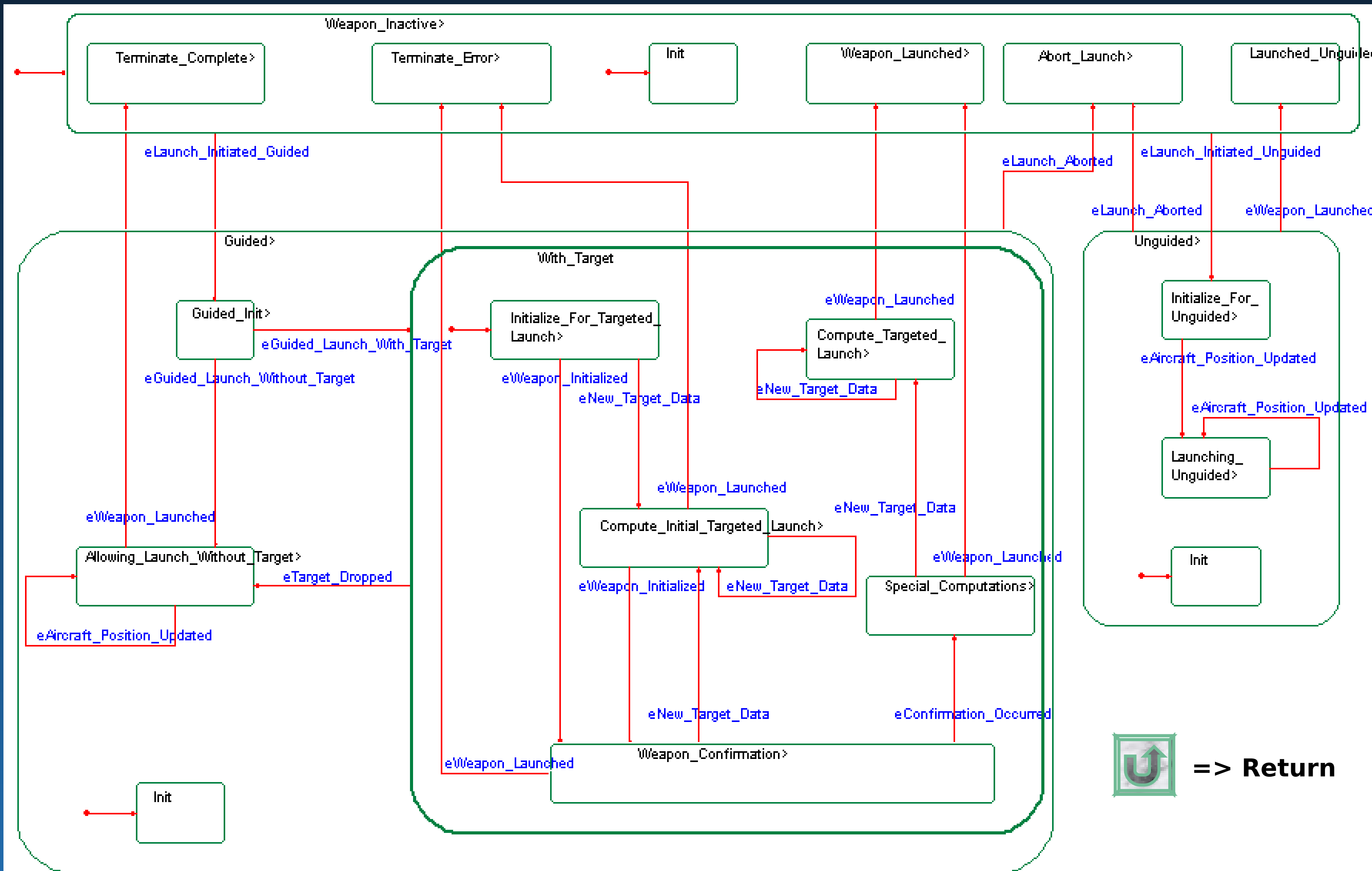
Instance Event Signups



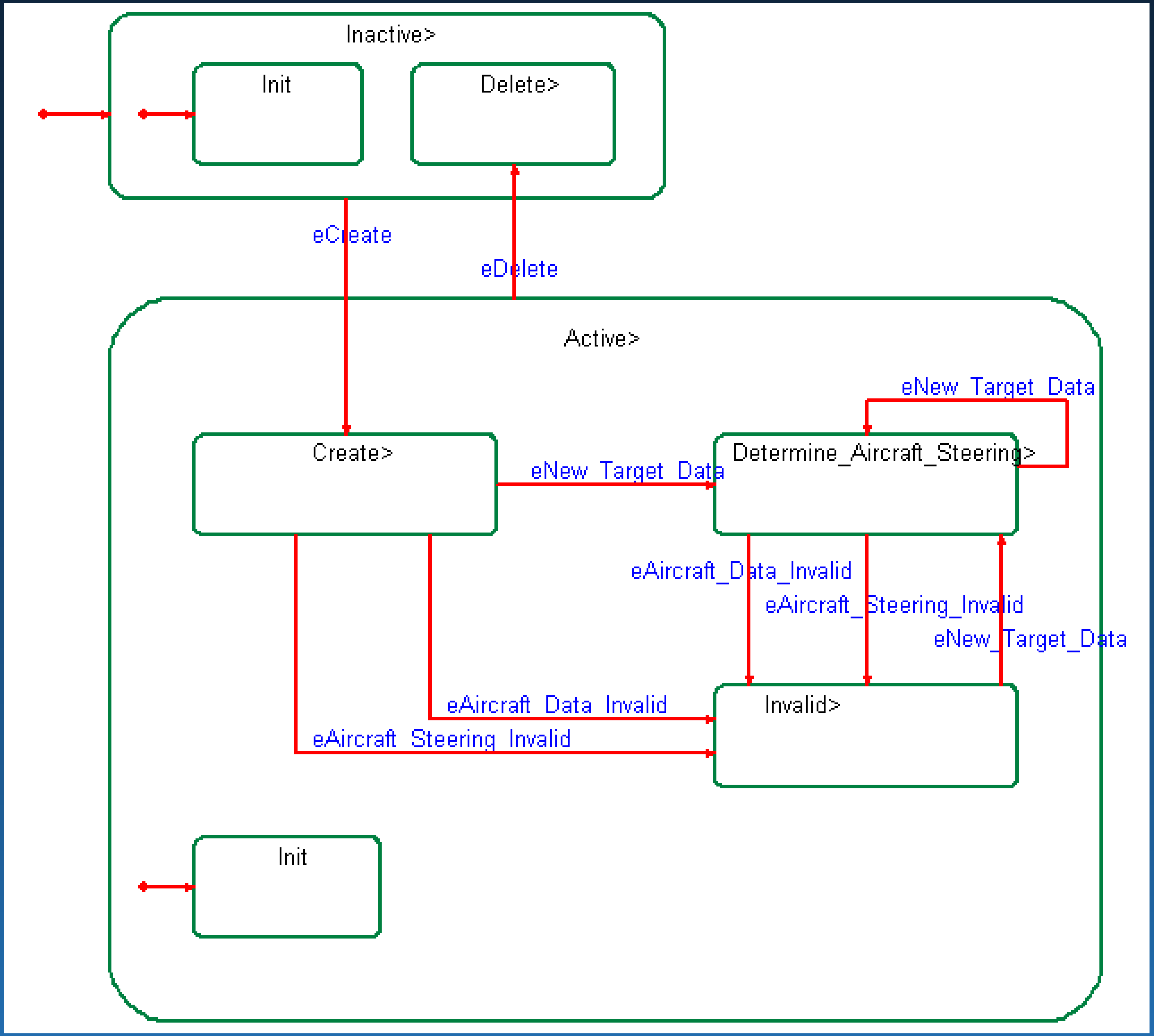

```
// Aircraft_Steering_Commands Constructor

// ...
// Sign up for eCreate (25Hz)
// Sign up for eNew_Target_Data (25 Hz)
// Sign up for aAircraft_Data_Invalid (25Hz)
// ...
// ...
```

=> Return

Weapon_Launch FSM

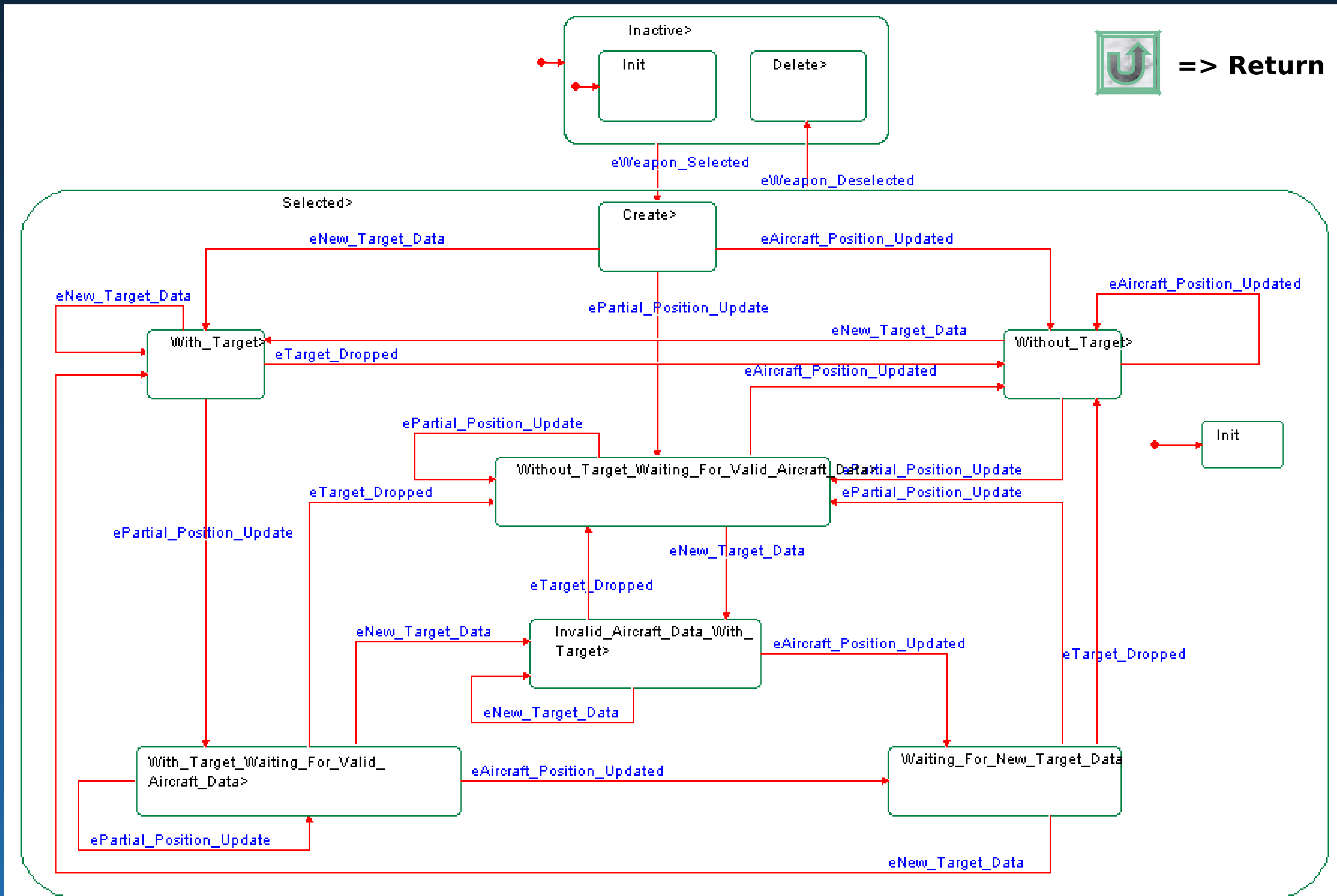


Aircraft_Steering_Commands FSM



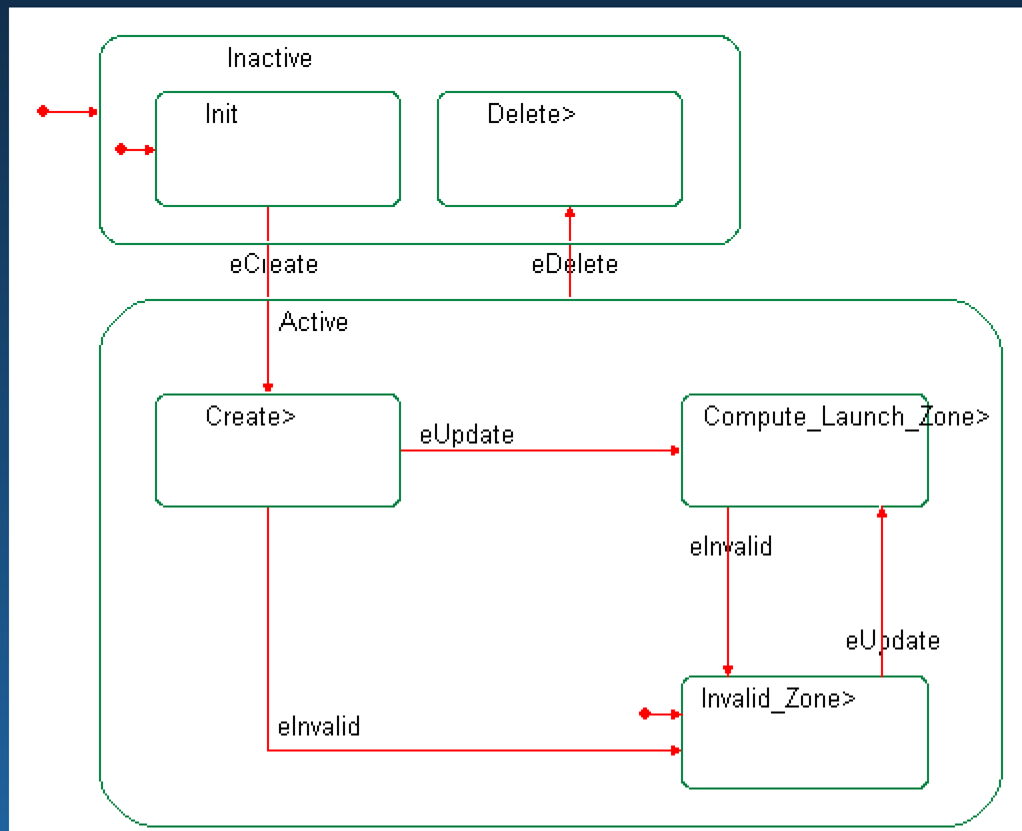
=> Return

Weapon_Pre_Launch FSM

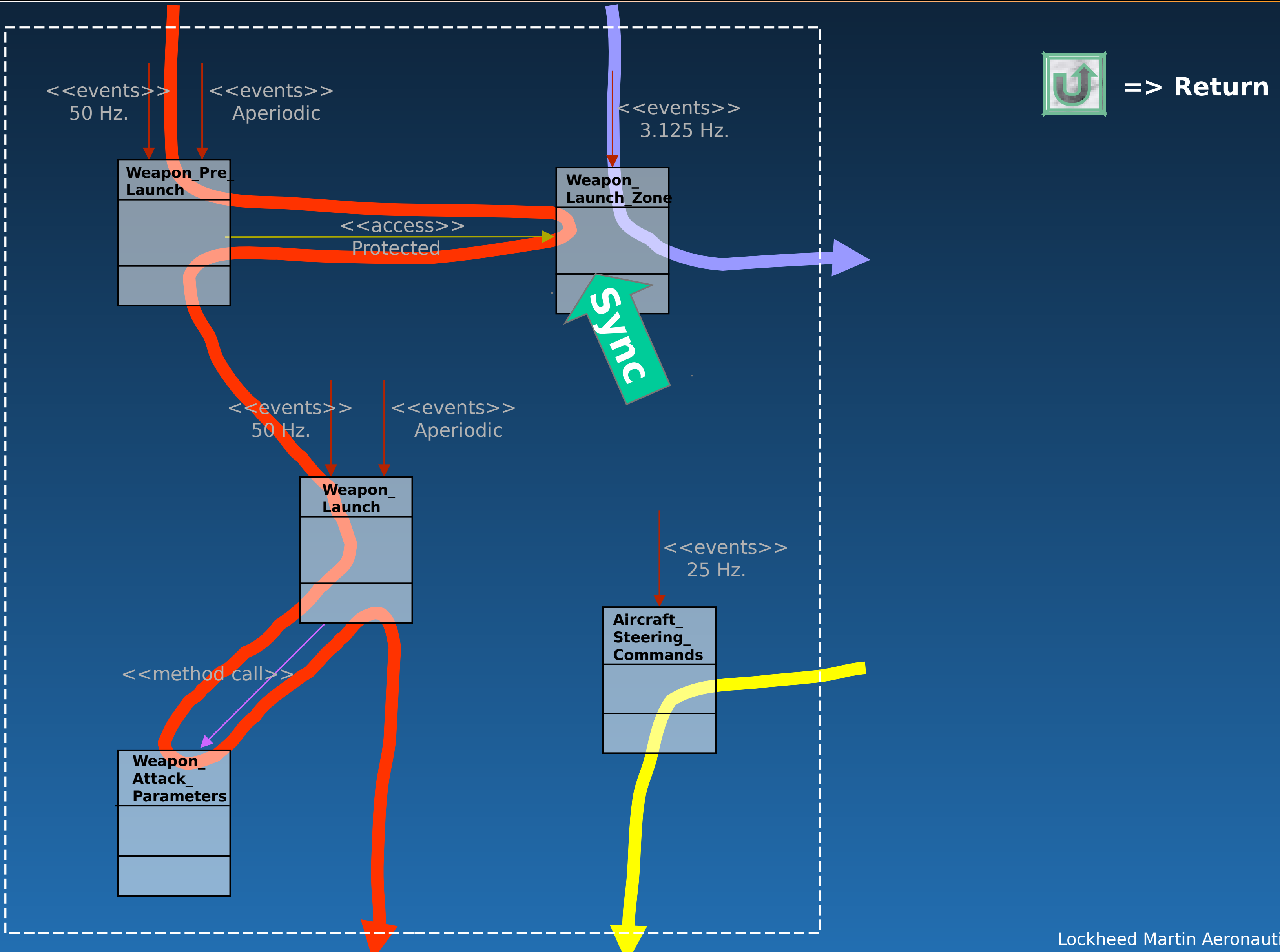


Weapon_Launch_Zone FSM

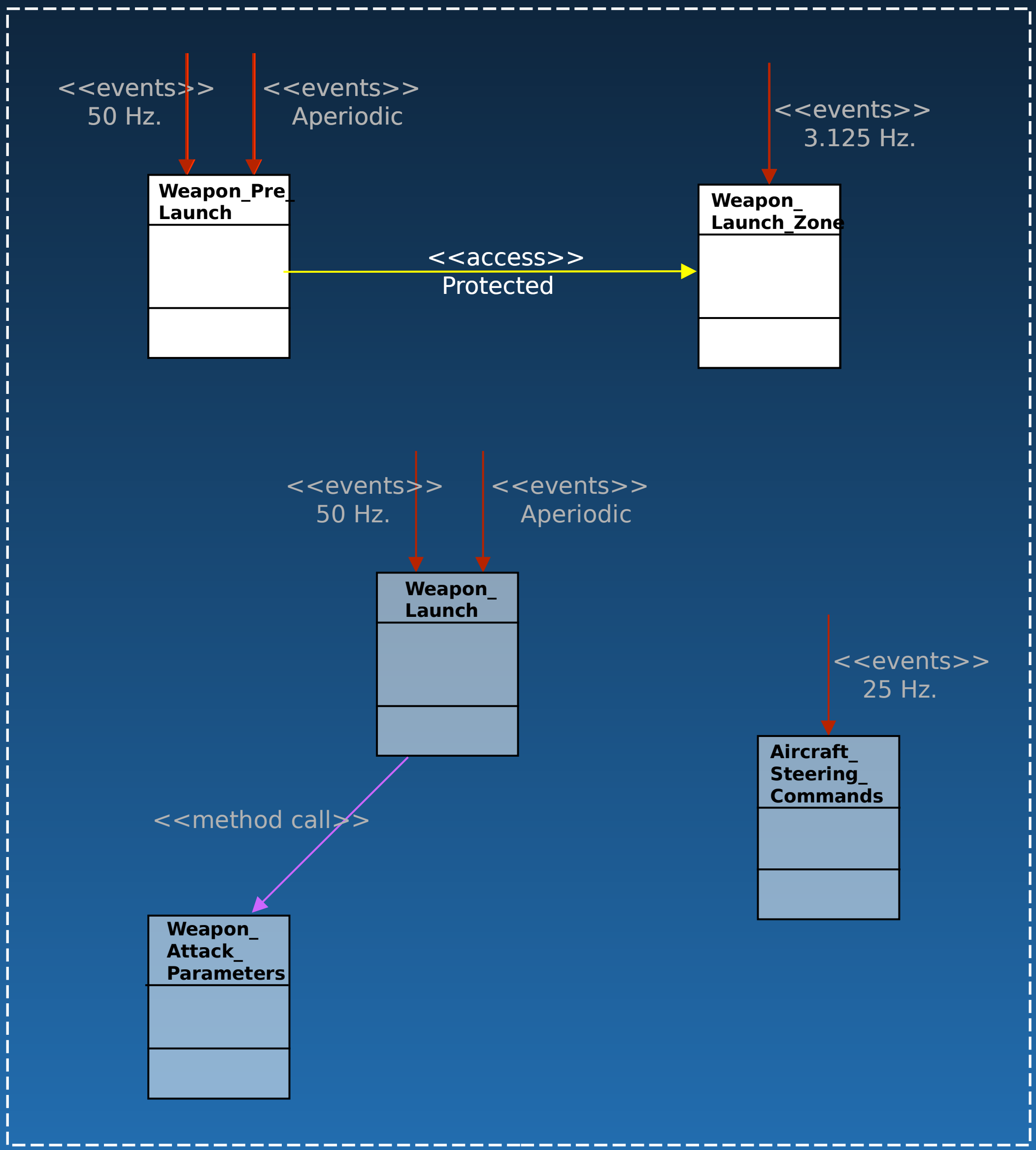
=> **Return**



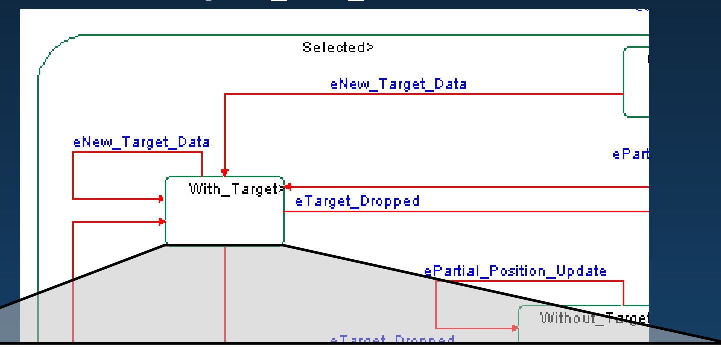
System Threads



Protected Data Access



Weapon_Pre_Launch FSM



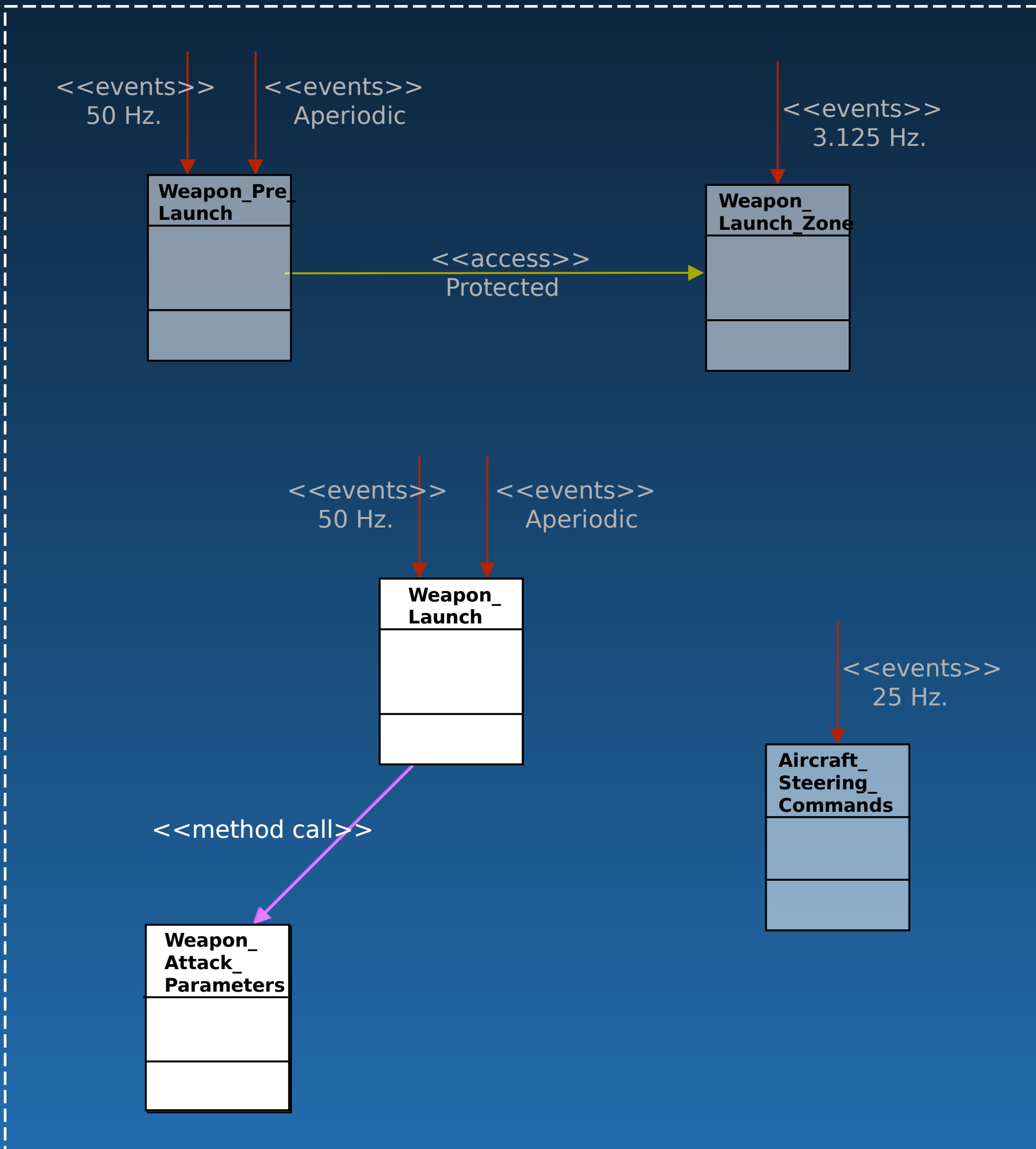

```
// aWith_Target

// …
theWeapon_Launch_Zone-
>Get_Zone_Data(Launch_Zone_Data);
// …
```

Weapon_Launch_Zone method

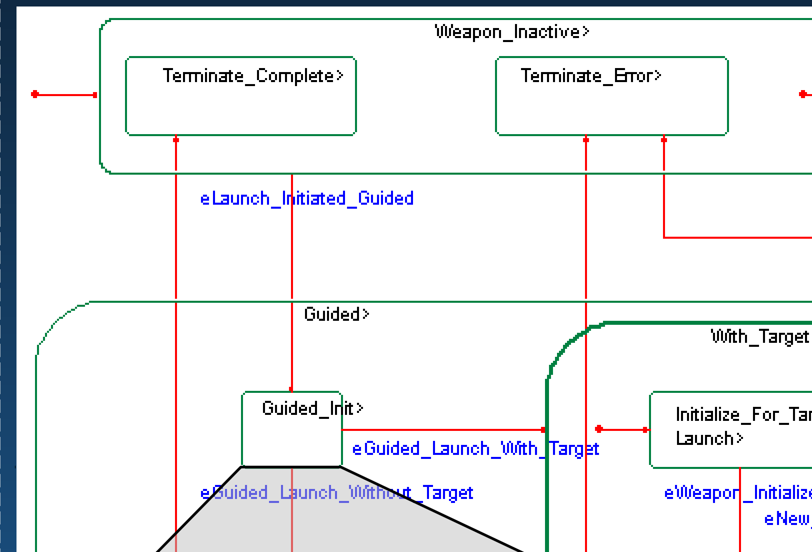
```
// Get_Zone_Data

startCriticalSection();

// Access zone data.
Launch_Zone_Data = a_Zone_Data;

stopCriticalSection();
```

=> Return

Computational Method Call



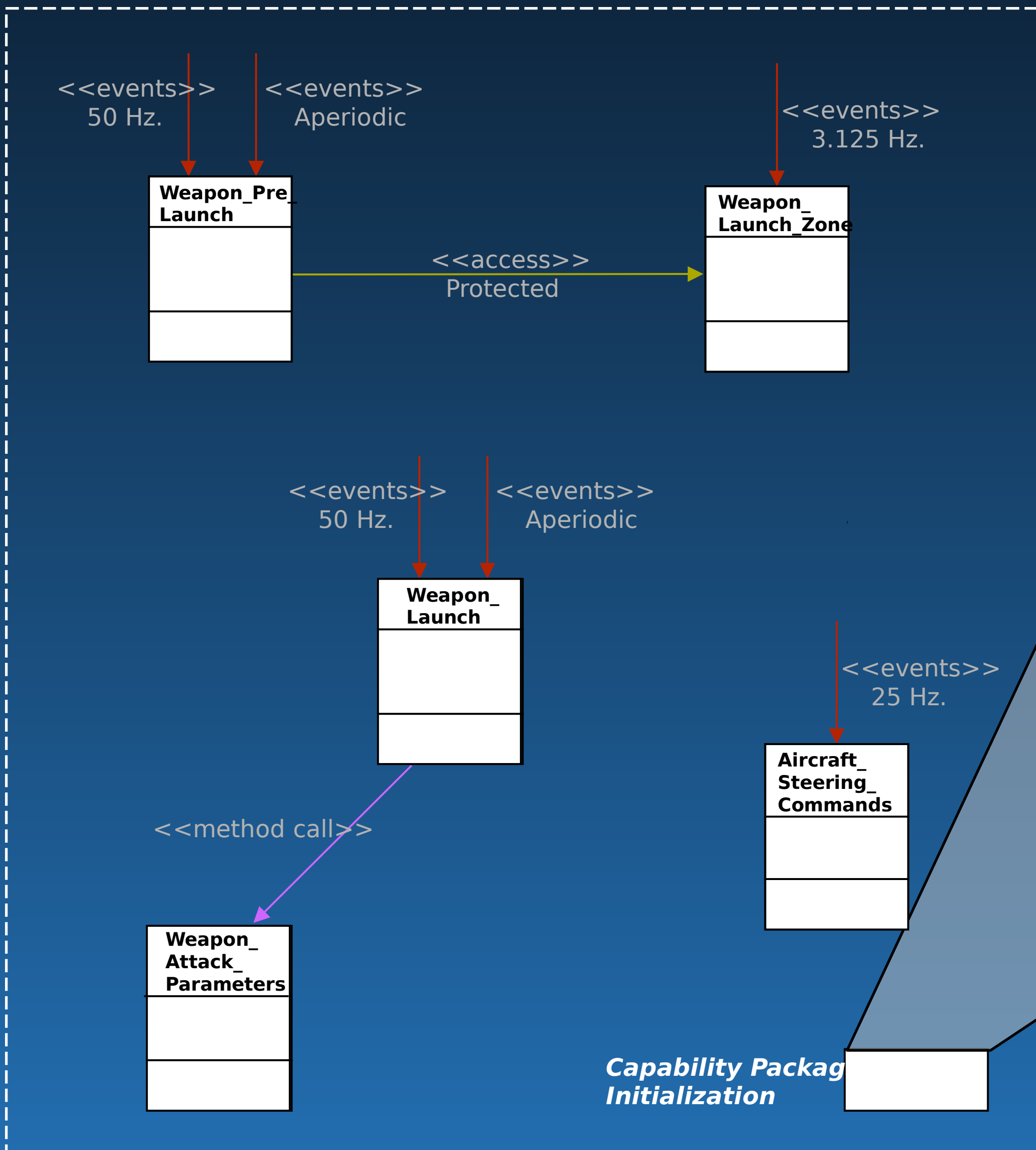
Weapon_Launch FSM




```
// aGuided_Init

// ...
// Example of an object to object computational method call
theWeapon_Attack_Parameters->Reset_Attack_Parameters;
// ...
```

=> Return

Aspect Instance Declarations




```
//  Aspect Instance Declarations

// ...
theWeapon_Attack_Parameters = new
Weapon_Attack_Parameters();
theAircraft_Steering_Commands = new
Aircraft_Steering_Commands();
theWeapon_Launch = new Weapon_Launch();
theWeapon_Pre_Launch = new
Weapon_Pre_Launch();
theWeapon_Launch_Zone = new
Weapon_Launch_Zone();
// ...
```

=> Return

Summary

- The STRIVE project plans to continue developing challenge problem characterizations and interacting with Phase 1 researchers
- We plan to continue representing DoD system needs and promoting transition of high payoff technologies
- Independent technology evaluations will also be performed
- We have the capability to perform in-context demonstrations using real and/or representative application software